INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2014; **98**:236–264
Published online 4 March 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.4624

Universal meshes: A method for triangulating planar curved domains immersed in nonconforming meshes

Ramsharan Rangarajan[1,*,†] and Adrián J. Lew[2]

[1]*School of Engineering, Brown University, 182 Hope Street, Providence RI 02912*
[2]*Department of Mechanical Engineering, Stanford University, CA 94305, USA*

SUMMARY

We introduce a new method to triangulate planar, curved domains that transforms a specific collection of triangles in a background mesh to conform to the boundary. In the process, no new vertices are introduced, and connectivities of triangles are left unaltered. The method relies on a novel way of parameterizing an immersed boundary over a collection of nearby edges with its closest point projection. To guarantee its robustness, we require that the domain be C^2-regular, the background mesh be sufficiently refined near the boundary, and that specific angles in triangles near the boundary be strictly acute. The method can render both straight-edged and curvilinear triangulations for the immersed domain. The latter includes curved triangles that conform *exactly* to the immersed boundary, and ones constructed with isoparametric mappings to interpolate the boundary at select points. High-order finite elements constructed over these curved triangles achieve optimal accuracy, which has customarily proven difficult in numerical schemes that adopt nonconforming meshes.

Aside from serving as a quick and simple tool for meshing planar curved domains with complex shapes, the method provides significant advantages for simulating problems with moving boundaries and in numerical schemes that require iterating over the geometry of domains. With no conformity requirements, the same background mesh can be adopted to triangulate a large family of domains immersed in it, including ones realized over several updates during the coarse of simulating problems with moving boundaries. We term such a background mesh as a *universal mesh* for the family of domains it can be used to triangulate. Universal meshes hence facilitate a framework for finite element calculations over evolving domains while using only fixed background meshes. Furthermore, because the evolving geometry can be approximated with any desired order, numerical solutions can be computed with high-order accuracy. We present demonstrative examples using universal meshes to simulate the interaction of rigid bodies with Stokesian fluids. Copyright © 2014 John Wiley & Sons, Ltd.

Received 15 January 2013; Revised 18 November 2013; Accepted 7 December 2013

KEY WORDS: universal meshes; immersed boundary; background meshes; mesh motion; moving boundary; curved finite elements

1. INTRODUCTION

We introduce a new way to triangulate sufficiently smooth planar, curved domains immersed in nonconforming triangulations. The method consists in mapping triangles in a given background mesh to (possibly) curvilinear ones resulting in a spatial discretization for the immersed domain that approximates it with *any* desired accuracy. Specifically, we show how to map select collections of triangles in a background mesh

(i) to a straight-edged triangulation of polygons whose sides interpolate the immersed boundary,
(ii) to a curvilinear triangulation whose curved edges conform *exactly* to the immersed boundary, and

*Correspondence to: Ramsharan Rangarajan, School of Engineering, Brown University, 182 Hope Street, Providence RI 02912.
†E-mail: ramsharan_rangarajan@brown.edu

(iii) to a curvilinear triangulation whose curved edges interpolate the immersed boundary at select points.

In addition to serving as a quick and simple tool for meshing planar domains with complex shapes, the method provides useful algorithmic advantages while simulating problems with moving boundaries and in numerical schemes that require iterating over the geometry of domains. We showcase the benefits of adopting curvilinear triangulations determined with this method to problems that are sensitive to boundary perturbations and in achieving optimal accuracy in high-order FEMs.

The idea presented here for mapping background meshes into conforming ones offers a fresh perspective on the familiar problem of triangulating/meshing planar domains [1–3]. We make specific choices for (1) the collection of triangles in the background mesh to be transformed and for (2) the mapping from these triangles onto a conforming triangulation for the domain. For the former, we select the triangles that have at least one vertex inside the domain. We then construct a mapping whose restriction to the boundary of the selected collection of triangles equals the closest point projection of the immersed boundary and that equals the identity beyond a small neighborhood of it. Simple modifications of such a construction yields straight-edged/curvilinear triangulations that interpolate the immersed boundary at select points rather than conform to it exactly, see Sections 2 and 4.

That said, the distinguishing feature of the method decidedly lies in the context of problems with evolving geometries. Such problems are ubiquitous, including ones with interactions between fluids and solids, problems with free boundaries and moving interfaces, domains with propagating cracks, problems involving phase transformations, and structural topology optimization. With universal meshes, we have addressed one important question that arises in their simulation, namely that of robustly discretizing the evolving domain. The reasons are quite conspicuous. Foremost among them is the fact that the *same* background mesh can be utilized for discretizing evolving domains by merely perturbing appropriate collections of triangles. We term such a background mesh a *universal mesh* for the family of domains it can be used to triangulate. A sufficiently refined triangulation of a square can hence serve as a universal mesh for a large family of domains immersed in it. Moreover, we impose no conformity requirements on the universal mesh; for instance, none of its vertices need to lie on the boundary. Connectivities of triangles are retained unaltered from the universal mesh and no new vertices are introduced. Hence, sparse patterns of data structures involved in numerical simulations can be retained across updates to the geometry. In Sections 6.2 and 6.3, we present demonstrative examples using universal meshes to simulate the interaction of rigid bodies with Stokesian fluids.

There is a large volume of literature on prescribing vertex motions for simulating problems with moving boundaries. For example, arbitrary Lagrangian–Eulerian methods displace vertices in the interior of moving domains based on the trajectories of vertices on moving boundaries and interfaces [4–10]. The requisite displacements/velocities for vertices are commonly computed using analogies with mass-spring systems [11, 12], hyper-elastic models [13, 14], or harmonic operators [15]. Vertex motion may also be prescribed to adapt meshes for the purpose of improving the accuracy of numerical simulations, as performed in r-adaptive methods [16–20].

In contrast to the above methods, we prescribe transformations for triangles in background meshes solely for the purpose of geometric conformity with the immersed boundary. Despite the seeming simplicity of such an idea, meshing algorithms based on just perturbing vertices are hardly ubiquitous in the literature. There is considerable freedom in deciding how to transform a background mesh into a conforming one. Our choice was briefly mentioned earlier; alternatives are certainly plausible. But the challenge lies in discerning when such algorithms are robust. It is common knowledge that perturbing vertices can result in degenerate or inverted triangles, and in general, yield tangled meshes, see Figure 3 for an example.

Without guarantees for the qualities of resulting meshes, heuristic perturbation-based meshing algorithms are seldom adopted in practice. Instead, evolving domains are commonly handled in numerical simulations by remeshing after each update to its boundary [21]. Immersed and embedded boundary methods on the other hand bypass the need for frequent remeshing by approximating domains using background meshes [22, 23]. However, these methods necessarily introduce the question of how to impose prescribed boundary conditions and interfacial constraints while accommodating nonconforming meshes. They may have to be imposed using penalty [24], with Lagrange multipliers [25, 26], with Nitsche's method and its variants [27, 28] or by enriching the space of admissible solutions near the immersed boundary as performed in extended FEMs [29, 30] and discontinuous Galerkin methods [31–33]. Phase field methods also enable tracking moving boundaries and interfaces over nonconforming meshes by augmenting problem formulations with additional fields [4, 34, 35].

Success of the method introduced here requires ensuring that the mapping defined over triangles in the background mesh onto the immersed domain is a homeomorphism. To this end, we have to impose restrictions on both the domain and the background mesh in which it is immersed. We assume sufficient regularity for the domain, restrict the local mesh size near its boundary, and insist that specific angles in triangles near the boundary be strictly acute. These restrictions are motivated and discussed in detail in Section 2.5. These conditions show for instance that any given smooth domain can be triangulated using only a sufficiently refined background mesh of equilateral triangles. We cannot however make such a claim with background meshes of right-angled triangles because such meshes may violate the requirement that certain angles be acute.

In addition to its simplicity and robustness, the triangulation method with universal meshes also renders spatial discretizations with high-order accuracy. Almost without exception, numerical schemes that adopt nonconforming background meshes, including the ones mentioned earlier, resort to polygonal approximations for the immersed domain. The accuracy of these schemes depends also on how prescribed boundary conditions and interfacial constraints are imposed. Consequently, achieving optimal accuracy with high-order interpolations has proven difficult. To construct high-order methods, it is imperative to approximate the immersed domain sufficiently well over the background mesh. Both constructions for curvilinear triangulations described in Section 4 achieve the requisite accuracy, see the example in Section 5.2. In fact, curved edges in the first construction conform exactly to the immersed boundary and thereby yields an *exact* discretization for it. The second is an isoparametric mapping that results from interpolating the first construction at select points of the background mesh. In Section 5.2, we revisit the problem of computing the deformation of a simply supported circular plate in bending, which is known to be sensitive to how well the curved boundary is represented [36].

Finally, we mention that the spatial discretization adopted for curved domains can prove critical not just to the accuracy of numerical schemes but also to their robustness. A familiar case in point are problems in computational contact mechanics. Adopting piecewise smooth (in particular polygonal) representations for smooth domains can result in nonsmooth contact events, which are often purely numerical artifacts that culminate in convergence difficulties [37, §9.6]. Such hindrances can instead be circumvented by resorting to the conforming curvilinear discretizations described here.

2. TRANSFORMING BACKGROUND MESHES INTO CONFORMING MESHES

It is both instructive and convenient to discuss the construction of straight-edged conforming triangulations from background meshes before proceeding to the construction of curvilinear ones. Consider a planar, bounded, curved domain Ω that is an open set in $\mathbb{R}^2$ and is immersed in a background triangulation $\mathcal{T}_h$. Recall that $\mathcal{T}_h$ is a triangulation if

(i) each triangle in $\mathcal{T}_h$ is a nonempty set and
(ii) if K_1 and K_2 are distinct triangles in $\mathcal{T}_h$, then $\overline{K}_1 \cap \overline{K}_2$ is either empty, a common edge or a common vertex.

The diameter of a triangle K in $\mathcal{T}_h$ is denoted by h_K, and the mesh size of $\mathcal{T}_h$ is denoted by the parameter $h := \max_{K \in \mathcal{T}_h} h_K$. By Ω being *immersed* in $\mathcal{T}_h$, we mean that the set triangulated by $\mathcal{T}_h$ contains $\overline{\Omega}$. We endow $\Gamma := \partial\Omega$ with an orientation specified by

$$\mathfrak{s}(x) := \begin{cases} -1, & \text{if } x \in \Omega, \\ +1 & \text{otherwise.} \end{cases} \tag{1}$$

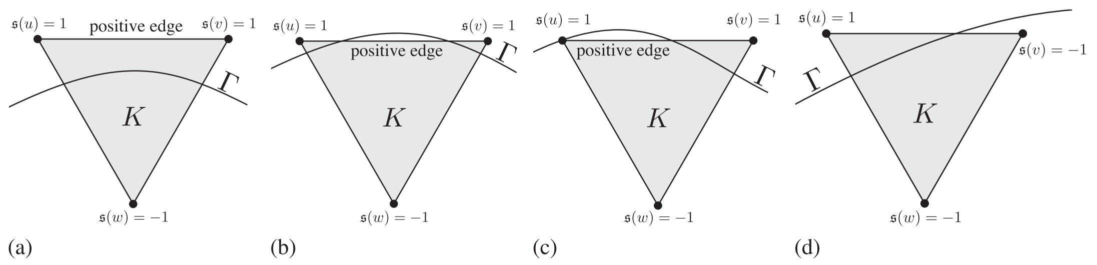


Figure 1. Examples illustrating the definition of triangles positively cut by Γ and their positive edges. In figures (a), (b), and (c), triangle K is positively cut because $\mathfrak{s}(u) = \mathfrak{s}(v) = 1$ and $\mathfrak{s}(w) = -1$. The edge joining vertices u and v is a positive edge. In figure (d), K is not positively cut because $\mathfrak{s} = 1$ at only one of its vertices.

The closest point projection onto Γ, $\pi : \mathbb{R}^2 \to \Gamma$, is defined as

$$\pi(x) := \arg\min_{y\in\Gamma} d(x, y), \tag{2}$$

where $d(\cdot,\cdot)$ is the Euclidean distance in $\mathbb{R}^2$. Related to the closest point projection is the signed distance to Γ, $\phi : \mathbb{R}^2 \to \mathbb{R}$ defined as $\phi(\cdot) := \mathfrak{s}(\cdot) \times d(\cdot,\Gamma)$, where $d(\cdot,\Gamma) := \min_{y\in\Gamma} d(\cdot, y)$.

At the outset, we assume no conformity between $\mathcal{T}_h$ and Γ. In particular, no vertex of $\mathcal{T}_h$ needs to lie on Γ. The mapping in (3) perturbs a few vertices of $\mathcal{T}_h$ in the vicinity of Γ to yield a triangulation $\mathcal{T}_h^c$ that conforms to Γ in the sense that the *collection of edges in* $\mathcal{T}_h^c$ *with both vertices on each connected component of* Γ *is nonempty and forms a simple, closed, polygonal chain.*[‡]

2.1. *Positively cut triangles, positive edges and conditioning angles*

To describe how to perturb vertices in the background mesh, we introduce the terminology of triangles *positively cut* by Γ. We say that a triangle K in $\mathcal{T}_h$ is positively cut by Γ if $\mathfrak{s} = +1$ at two vertices and $\mathfrak{s} = -1$ at the remaining one. In Figures 1a, 1b, and 1c, K with vertices $\{u, v, w\}$ is positively cut by Γ because $\mathfrak{s}(u) = \mathfrak{s}(v) = 1$ and $\mathfrak{s}(w) = -1$. We then call the edge joining vertices u and v as a *positive edge* with respect to Γ. When the curve Γ is clear from the context, we will simply refer to positively cut triangles and positive edges. The union of positive edges in $\mathcal{T}_h$ is denoted by Γ_h. Identifying positively cut triangles and positive edges in a background mesh requires only examining the value of $\mathfrak{s}$ at its vertices. It does not, for instance, entail inspecting how or how many times Γ intersects the edges in the background mesh.

The *proximal vertex* of a positively cut triangle K is the vertex of its positive edge closest to Γ. In Figure 1c for example, u is the proximal vertex of K. When both vertices of the positive edge are equidistant from Γ, the one containing the smaller interior angle of K is designated to be the proximal vertex. If the angles are equal as well, either vertex of the positive edge can be designated as the proximal vertex. Finally, the *conditioning angle* of a positively cut triangle is the interior angle at its proximal vertex.

2.2. *Description of the meshing algorithm*

Let $\mathcal{T}_h^{\text{sub}}$ be the collection of triangles in $\mathcal{T}_h$ that have $\mathfrak{s} = -1$ at one or more vertices. The meshing algorithm consists in transforming $\mathcal{T}_h^{\text{sub}}$ to $\mathcal{T}_h^c$ and is succinctly summarized as the mapping M_h defined over vertices in $\mathcal{T}_h^{\text{sub}}$ as

$$M_h(x) = x - f_h(x)\, N(\pi(x)), \tag{3}$$

[‡]A polygonal chain is a curve specified by a sequence of points $(v_0, v_1, \ldots, v_n)$ called its vertices, so that the curve consists of line segments connecting consecutive vertices.

where N is the unit outward normal to Γ. The function f_h in (3) is defined using parameters $\alpha \in (0, 1]$ and $R_r > h$ equal to a few multiples of the mesh size as

$$f_h(x) = \begin{cases} \phi(x) & \text{if } x \in \Gamma_h, \\ \alpha h \max\left\{0, 1 + \dfrac{\phi(x)}{R_r}\right\} & \text{otherwise.} \end{cases} \tag{4}$$

We will postpone a discussion of when M_h is well defined and instead examine a few critical features evident from (3) and (4). It is clear from (3) that M_h perturbs vertices along the direction of the local normal to Γ by a distance that is modulated by f_h. Examining the action of M_h on positive edges, we see that

$$x \in \Gamma_h \;\Rightarrow\; f_h(x) = \phi(x) \;\Rightarrow\; M_h(x) = x - \phi(x)\, N(\pi(x)) = \pi(x), \tag{5}$$

where the last equality holds when Γ is sufficiently smooth and x lies close to it [38]. Hence, vertices of positive edges are snapped to their closest point in Γ.

To accommodate such snapping, vertices in the neighborhood $B(\Gamma, R_r) := \{x \in \mathbb{R}^2 : d(x, \Gamma) < R_r\}$ are relaxed away from Γ. At vertices in Ω farther than distance R_r from Γ, we have

$$x \in \Omega \setminus B(\Gamma, R_r) \;\Rightarrow\; \phi(x) < -R_r \;\Rightarrow\; f_h(x) = 0 \;\Rightarrow\; M_h(x) = x. \tag{6}$$

Hence, vertices in Ω that lie outside the R_r-neighborhood of Γ are left undisturbed by M_h. Since R_r is chosen to be a few multiples of the mesh size, (6) indeed shows that M_h is a *local* perturbation of vertices near Γ. It is also a *small* perturbation because the distance by which these vertices are perturbed equals f_h and $|f_h| \leqslant \alpha h$.

The specific form of f_h in (4) is one that is particularly suitable for relaxing vertices in background meshes that have uniformly sized triangles near the boundary. It leaves much to be desired—notice that f_h is in fact (and in general) discontinuous at Γ_h. This is not a problem because as we will see in Section 2.5, we only need control over the magnitude of f_h and its difference quotient over vertices near Γ. There is certainly room to improve the choice of f_h, especially to relax vertices by distances that are commensurate with perturbations of vertices of positive edges in background meshes that have triangles of widely varying sizes near the boundary.

2.3. *An illustrative example*

Figure 2 shows an example of a curved domain bounded by a collection of 22 cubic splines immersed in a background mesh of equilateral triangles. For the choice $R_r = 3h$, Table I lists some statistics on the resulting conforming meshes. Starting with a background mesh $\mathcal{T}_h$, the meshes refine($\mathcal{T}_h$), refine^2($\mathcal{T}_h$), and refine^3($\mathcal{T}_h$) in the table correspond to its successive self-similar refinements. With each such refinement, the mesh size is halved and the number of triangles is quadrupled.

Table I highlights that most triangles in the conforming mesh are retained unaltered from the background mesh, especially when the latter is sufficiently refined. For the choice $\alpha = 0.75$, qualities of triangles in the conforming meshes are reported in Table IIa. Besides the minimum and maximum angles, we also report the ratio of the circumradius to the inradius, which has a best possible value of 2 that is attained in equilateral triangles. Table IIb incorporates additional perturbations for a few vertices close to the boundary to improve the quality of the images of positively cut triangles in the conforming mesh. Details of these perturbations are described subsequently in Section 3.5. The meshes in Figure 2 correspond to the second row in Tables I and II. Table II serves to show that the quality and extreme angles of the computed meshes for the example do not deteriorate as the background mesh is refined.

2.4. *Need for restrictions on background meshes*

Perturbing vertices according to M_h in (3) raises a few inescapable questions.

(i) Suppose that M_h is well defined and the vertex perturbation algorithm executes successfully. Is the resulting mesh $\mathcal{T}_h^c$ a conforming triangulation of Ω?

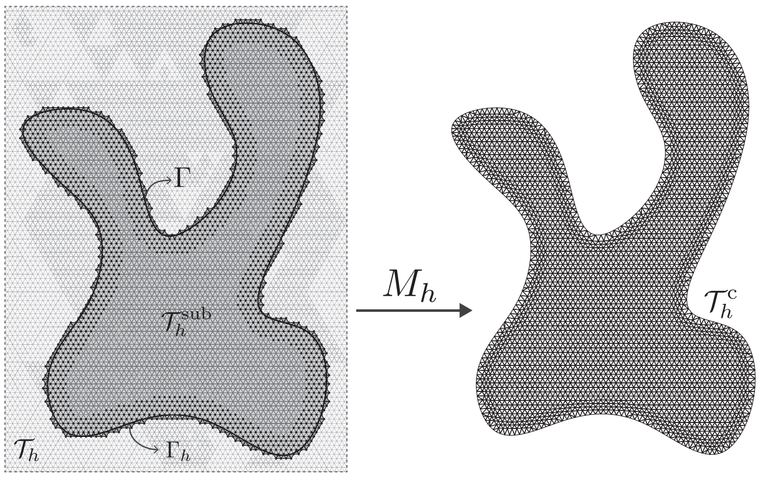


Figure 2. Example demonstrating the algorithm to perturb vertices of a collection of triangles in a background mesh to make it conform to a curved domain. The domain shown on the left is enclosed by a collection of cubic splines and is immersed in a mesh of equilateral triangles $\mathcal{T}_h$. The triangles shaded in light and dark gray have at least one vertex in the domain and constitute the sub-triangulation $\mathcal{T}_h^{\rm sub}$. Triangles positively cut by the boundary are shaded in dark gray. Just the vertices marked by dark dots are perturbed by M_h, thereby transforming $\mathcal{T}_h^{\rm sub}$ into the conforming mesh $\mathcal{T}_h^c$ shown on the right.

Table I. Given a background mesh $\mathcal{T}_h$, refine^i $(\mathcal{T}_h)$ is the mesh resulting from i self-similar subdivisions of $\mathcal{T}_h$. With $\mathcal{T}_h$ and its successive refinements as background meshes, the second column in the table lists the number of triangles that are retained in the conforming mesh for the domain shown in Figure 2. The number of triangles positively cut by the boundary are listed in the third column. Column four lists the fraction of triangles in the conforming mesh that are retained unaltered (equilateral) from the background mesh for the choice $R_r = 3h$. Since $\mathcal{T}_h$ is quite coarse, a significant fraction of triangles in $\mathcal{T}_h^{\rm sub}$ are altered in the algorithm; with refinement, we see that most triangles in the background mesh are left undisturbed.

	# transformed	# positively cut	% unaltered
$\mathcal{T}_h$	1808	180	24.6
refine$(\mathcal{T}_h)$	6874	356	58.3
refine$^2(\mathcal{T}_h)$	26744	708	78.5
refine$^3(\mathcal{T}_h)$	105525	1415	89.0

Table II. Qualities of triangles in conforming meshes in the example discussed in Section 2.3 while using parameters $R_r = 3h$ and $\alpha = 0.75$ in (4). Ranges of three quality metrics are listed— the maximum value of the ratio of the circumradius to the inradius is reported in the first column and the minimum and maximum angles in the mesh are reported in subsequent columns. Table IIb incorporates additional perturbations for vertices close to the boundary that alters the collection of positively cut triangles to improve the quality of their images. Details of these perturbations are given in Section 3.5. These perturbations require a parameter η, which is chosen to be 0.2. We highlight the resulting improvement in qualities of the meshes in Table IIb compared to the ones in Table IIa.

(a)

	max. quality	min. angle	max. angle
$\mathcal{T}_h$	4.45	20.8°	119.3°
refine$(\mathcal{T}_h)$	4.45	23.2°	118.9°
refine$^2(\mathcal{T}_h)$	4.99	20.4°	121.9°
refine$^3(\mathcal{T}_h)$	5.05	20.3°	122.3°

(b)

	max. quality	min. angle	max. angle
$\mathcal{T}_h^*$	4.43	20.6°	117.3°
refine$(\mathcal{T}_h)^*$	4.88	19.1°	118.9°
refine$^2(\mathcal{T}_h)^*$	4.26	22.2°	115.5°
refine$^3(\mathcal{T}_h)^*$	4.34	21.4°	115.5°

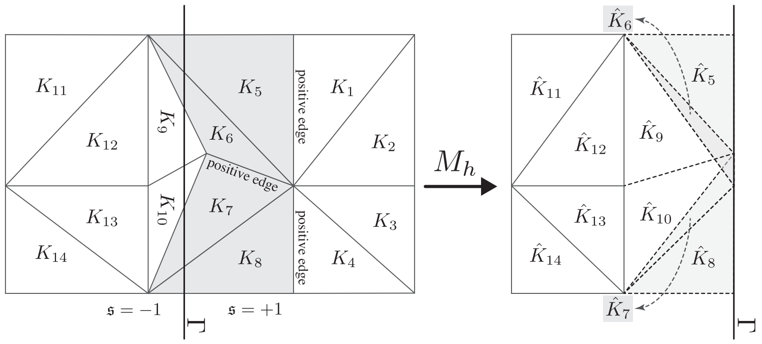


Figure 3. Example to illustrate that without restrictions on the background mesh, perturbing vertices with M_h in (3) can result in degenerate and overlapping triangles even when the boundary Γ is locally straight. For simplicity, only a few triangles near the boundary are shown. Triangles K_{5-14} belong to $\mathcal{T}_h^{\text{sub}}$, and K_{5-8} shaded in gray are positively cut by Γ. Upon perturbing the vertices according to (3), the images of triangles K_{5-10} overlap in the resulting mesh, and the orientations of triangles K_6 and K_7 are reversed. Since Γ is locally straight, this example shows that merely refining the background mesh cannot ensure success of the meshing algorithm; additional restrictions are necessary.

(ii) When is the mapping M_h well defined? If not always, can we find sufficient conditions for M_h to be well defined and for $\mathcal{T}_h^c$ to be a conforming triangulation of Ω?

Neither question is pedagogical. Answering them is essential for practical implementations and to guarantee the qualities of resulting meshes.

It is in fact quite easy to concoct examples that answer the first question —ones in which M_h is well defined but the resulting mesh is tangled, see Figure 3. For simplicity, the boundary in the figure is depicted as being (locally) straight. Only a subset of it is shown along with the nearby triangles in the background mesh. In the example, triangles $K_5 - K_8$ shaded in gray are positively cut. Projecting vertices of their positive edges onto the boundary cause the images of triangles K_{5-10} to overlap in the resulting mesh in addition to reversing the orientations of triangles K_6 and K_7.

It is therefore evident that unless we impose restrictions on the background mesh, perturbing its vertices may result in degenerate, inverted, and overlapping triangles of arbitrarily poor quality even when the boundary has no curvature or features. Since Γ in the example shown has no inherent length scale, merely refining the background mesh cannot guarantee success of the algorithm. While considering curved boundaries with finite feature sizes, it is also necessary to consider sufficiently refined background meshes. We refer the reader to [39] for more examples complementing the discussion here; we do not repeat them here for the sake of brevity.

2.5. *Guaranteeing success of the meshing algorithm: a simplified case*

To guarantee that M_h is well defined and perturbing vertices in $\mathcal{T}_h^{\text{sub}}$ according to M_h yields a valid triangulation $\mathcal{T}_h^c$ over Ω with bounded quality, the following conditions suffice:

(i) the domain Ω is C^2-regular,
(ii) Ω is immersed in $\mathcal{T}_h$ in the sense that $\overline{\Omega} \subset \cup_{K \in \mathcal{T}_h} \overline{K}$,
(iii) the triangulation $\mathcal{T}_h$ is sufficiently refined in the vicinity of Γ,
(iv) conditioning angle in each positively cut triangle in $\mathcal{T}_h$ is strictly smaller than 90°. Conditions (i)–(iv) guarantee that $\pi : \Gamma_h \to \Gamma$ is a homeomorphism. To ensure that the mapping for relaxing vertices is robust, we additionally require that
(v) triangles with precisely two vertices in Ω are acute-angled,
(vi) parameter $R_r > h$ equals a few multiples of h, and
(vii) parameter α is chosen such that $(1 + h/R_r)^{-1} \leqslant \alpha \leqslant 1$.

A precise definition of C^2-regular domains can be found in [38]; for our discussion, it suffices to note that Ω is C^2-regular if the signed distance ϕ to Γ is C^2 in a neighborhood of Γ.

Consider first the question of when M_h is well defined. Clearly, f_h in (4) is well defined because $R_r \neq 0$. Following [38], we note that a C^2-regular boundary Γ has a well defined normal N (in fact differentiable) because it is a C^2 curve. There is also a radius $r_n > 0$, smaller than or equal to the distance from the medial axis, such that π is C^1 and ϕ is C^2 in the neighborhood $B(\Gamma, r_n)$. By definition, points in Γ_h are within distance h from Γ. Hence, when h is sufficiently small, $\Gamma_h \subset \overline{B(\Gamma, h)} \subset B(\Gamma, r_n)$ and consequently M_h is well defined because each term appearing in (3) is.

Demonstrating that assumptions (i)-(vii) suffice to guarantee that the mesh $\mathcal{T}_h^c$ is a valid one is less trivial. It entails showing that the transformation of triangles in $\mathcal{T}_h^{\text{sub}}$ to $\mathcal{T}_h^c$ is a homeomorphism. In the following, we examine the special case in which Γ is locally straight. Although quite simplistic, the calculations shown reveal the essential details, particularly the need for certain angles to be acute and the choice of the relaxation map in (4).

The calculations shown next consider only the case $h/r_n \ll 1$. Consequently, we lose the opportunity to examine the requirements on the mesh size that arise from the curvature and feature sizes of the boundary. At the end of this section, we comment on the additional details that need to be considered when Γ has nonzero curvature and finite feature sizes (i.e., when h/r_n is small but finite).

2.5.1. Notation. Let (x, y) be a Cartesian coordinate system for $\mathbb{R}^2$ such that $\Gamma = \{(x, y) : y = 0\}$ and $\phi(x, y) = y$. Let triangle K in $\mathcal{T}_h$ have vertices $\{u_1, u_2, u_3\}$ and let vertex u_i have coordinates (x_i, y_i) for $i = 1, 2, 3$. We choose an orientation for K such that its signed area Δ_K

$$\Delta_K = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} > 0. \tag{7}$$

Denote the lengths of edges joining the pairs of vertices u_1 to u_2, u_2 to u_3, and u_3 to u_1 by ℓ_1, ℓ_2, and ℓ_3 respectively, and let $\tau_i := h/\ell_i$.

We examine the quality of the image $\tilde{K}$ of K that has vertices $\{\tilde{u}_1, \tilde{u}_2, \tilde{u}_3\}$, where $\tilde{u}_i := M_h(u_i)$ for $i = 1, 2, 3$. We consider the cases in which one, two, or all three vertices of K lie in $B(\Gamma, R_r) \cap \Omega$. We show that $\tilde{K}$ has nonzero area $\Delta_{\tilde{K}}$, the same orientation as K, and compute bounds for its quality. As the quality metric Q, we pick one that is easy to estimate—the ratio of the longest to the shortest edge. This non-dimensional metric has a best possible value of 1, attained in equilateral triangles. Recall that in Table II, we used the ratio of the circumradius and the inradius to examine the qualities of triangles. Alternative quality metrics and their properties can be found in [40]. Notice that although the metric Q does not help identify flat triangles, together with the lower bound for $\Delta_{\tilde{K}}$, it in fact shows that flat triangles do not occur in the conforming mesh $\mathcal{T}_h^c$ and guarantees bounds for the ratio of the circumradius to the inradius.

Finally, the lengths of edges in $\tilde{K}$ are denoted by $\tilde{\ell}_i, i = 1, 2, 3$, defined analogous to ℓ_i's. The factor $\mu := (1 - \alpha h/R_r)$, which appears repeatedly in the following calculations, lies between 0 and 1 owing to the choices $\alpha < 1$ and $R_r > h$.

2.5.2. Triangles with all vertices in $\Omega \cap B(\Gamma, R_r)$. By definition of M_h, we have

$$\tilde{u}_i = \left(x_i, y_i - \alpha h\left(1 + \frac{y_i}{R_r}\right)\right). \tag{8}$$

Then

$$\Delta_{\tilde{K}} = \frac{1}{2}\begin{vmatrix} 1 & x_1 & y_1 - \alpha h\left(1 + \frac{y_1}{R_r}\right) \\ 1 & x_2 & y_2 - \alpha h\left(1 + \frac{y_2}{R_r}\right) \\ 1 & x_3 & y_3 - \alpha h\left(1 + \frac{y_3}{R_r}\right) \end{vmatrix} = \frac{1}{2}\left(1 - \frac{\alpha h}{R_r}\right)\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} = \mu \Delta_K. \tag{9}$$

It follows that $\tilde{K}$ has nonzero measure and identical orientation as K. The length $\tilde{\ell}_1$ of the edge joining $\tilde{u}_1$ and $\tilde{u}_2$ is

$$\tilde{\ell}_1^2 = (x_1 - x_2)^2 + \left(y_1 - y_2 - \frac{\alpha h}{R_r}(y_1 - y_2)\right)^2 = \ell_1^2 - \frac{\alpha h}{R_r}\left(2 - \frac{\alpha h}{R_r}\right)(y_1 - y_2)^2. \tag{10}$$

Using $0 \leqslant |y_1 - y_2| \leqslant \ell_1$ in (10), we obtain

$$\mu \ell_1 \leqslant \tilde{\ell}_1 \leqslant \ell_1. \tag{11}$$

Analogous estimates hold for lengths $\tilde{\ell}_2$ and $\tilde{\ell}_3$, resulting in the estimate for the quality of $\tilde{K}$ as

$$\mu \leqslant \frac{Q(\tilde{K})}{Q(K)} \leqslant \frac{1}{\mu}. \tag{12}$$

2.5.3. Triangles with exactly two vertices in Ω. Without loss of generality, assume that $u_1 \notin \Omega$. Hence, $\tilde{u}_1 = (x_1, 0)$ while the coordinates of $\tilde{u}_2$ and $\tilde{u}_3$ are given by (8). Similar to the computation performed earlier, we have

$$\Delta_{\tilde{K}} = \frac{1}{2}\begin{vmatrix} 1 & x_1 & 0 \\ 1 & x_2 & y_2 - \alpha h\left(1 + \frac{y_2}{R_r}\right) \\ 1 & x_3 & y_3 - \alpha h\left(1 + \frac{y_3}{R_r}\right) \end{vmatrix} = \mu \Delta_K + \frac{1}{2}(x_3 - x_2)(\alpha h - \mu y_1) \tag{13}$$

In the following, we use (7) and the assumption that K is acute-angled to show $x_3 \geqslant x_2$. Then, because $y_1 \leqslant h \Rightarrow \alpha h - \mu y_1 \geqslant 0$ when $\alpha \geqslant 1/(1 + h/R_r)$, (13) yields the lower bound $\Delta_{\tilde{K}} \geqslant \mu \Delta_K$. The upper bound follows immediately from $y_1 \geqslant 0$ and $|x_3 - x_2| \leqslant \ell_2 \leqslant h$. Pending the demonstration of $x_3 \geqslant x_2$, we obtain bounds for $\Delta_{\tilde{K}}$ as

$$\mu \Delta_K \leqslant \Delta_{\tilde{K}} \leqslant \mu \Delta_K + \frac{1}{2}\alpha h^2. \tag{14}$$

Now, let us see that $x_3 \geqslant x_2$. To this end, suppose that $x_3 < x_2$. Using $x_3 < x_2$ and $y_2, y_3 < 0 \leqslant y_1$ in (7), we obtain

$$\begin{aligned} 2\Delta_K &= y_1(x_3 - x_2) + y_2(x_1 - x_3) + y_3(x_2 - x_1), \\ &\leqslant y_2(x_1 - x_3) + y_3(x_2 - x_1), \\ &= |y_2|(x_3 - x_1) + |y_3|(x_1 - x_2). \end{aligned} \tag{15}$$

Equation (15) reveals that $y_2 < y_3$ is a necessary condition for $\Delta_K > 0$. It additionally shows that $x_1 < x_3$ because both $x_3 \leqslant x_1 \leqslant x_2$ and $x_3 < x_2 \leqslant x_1$ yield $\Delta_K \leqslant 0$. Now, from the assumption that the interior angle at vertex u_3 is acute, we know $(u_1 - u_3) \cdot (u_2 - u_3) > 0$ yielding

$$(x_1 - x_3)(x_2 - x_3) + (y_1 - y_3)(y_2 - y_3) > 0. \tag{16}$$

The first term in (16) is nonpositive because $x_1 < x_3 < x_2$ while the second term is nonpositive because $y_2 < y_3 < y_1$. In this way, we conclude that $x_2 > x_3$ is impossible, which in turn affirms the lower bound in (14).

Next, to compute the quality of $\tilde{K}$, we estimate the lengths of its edges. We have

$$\tilde{\ell}_1^2 = (x_1 - x_2)^2 + \left(y_2 - \alpha h\left(1 + \frac{y_2}{R_r}\right)\right)^2. \tag{17}$$

Using $0 \leqslant |x_1 - x_2| < \ell_1$ and $-\ell_1 < y_2 < 0$ in (17), we obtain

$$\alpha^2 h^2 \leqslant \tilde{\ell}_1^2 \leqslant \ell_1^2 + (\alpha h + \mu \ell_1)^2. \tag{18}$$

Thus we have shown that

$$\alpha \tau_i \ell_i \leqslant \tilde{\ell}_i \leqslant \sqrt{1 + (\alpha \tau_i + \mu)^2}\,\ell_i \quad \text{for } i = 1, 3, \tag{19}$$

while as computed in (11), we have

$$\mu \ell_2 \leqslant \tilde{\ell}_2 \leqslant \ell_2. \tag{20}$$

Hence, the quality metric for $\tilde{K}$ is estimated by

$$\frac{\min\{\alpha \min_i \tau_i, \mu\}}{\sqrt{1 + (\alpha \max_i \tau_i + \mu)^2}} \leqslant \frac{Q(\tilde{K})}{Q(K)} \leqslant \frac{\sqrt{1 + (\alpha \max_i \tau_i + \mu)^2}}{\min\{\alpha \min_i \tau_i, \mu\}}. \tag{21}$$

2.5.4. Triangles with exactly one vertex in Ω (positively cut triangles). Next, we consider how positively cut triangles are transformed. Let K be positively cut, assume that the edge joining vertices u_1 and u_2 is its positive edge and that u_1 is its proximal vertex. Hence

$$y_2 \geqslant y_1 \geqslant 0 > y_3. \tag{22}$$

In the following, we first estimate the length of the edge joining vertices $\tilde{u}_1$ and $\tilde{u}_2$, namely $\tilde{\ell}_1 = |x_1 - x_2|$. Specifically, we show that the acute conditioning angle assumption helps bound $\tilde{\ell}_1$ away from zero. The idea behind the calculation is illustrated in Figure 4.

Since the conditioning angle ϑ_K of triangle K, namely the interior angle at vertex u_1, is strictly smaller than 90°, we have

$$0 < \cos \vartheta_K := \frac{(u_2 - u_1)}{\ell_1} \cdot \frac{(u_3 - u_1)}{\ell_3} = \frac{(x_2 - x_1)(x_3 - x_1) + (y_2 - y_1)(y_3 - y_1)}{\ell_1 \ell_3}. \tag{23}$$

Noting from (22) that $(y_2 - y_1)(y_3 - y_1) \leqslant 0$, (23) implies that $(x_2 - x_1)(x_3 - x_1) > 0$. Then, without loss of generality, let us assume that

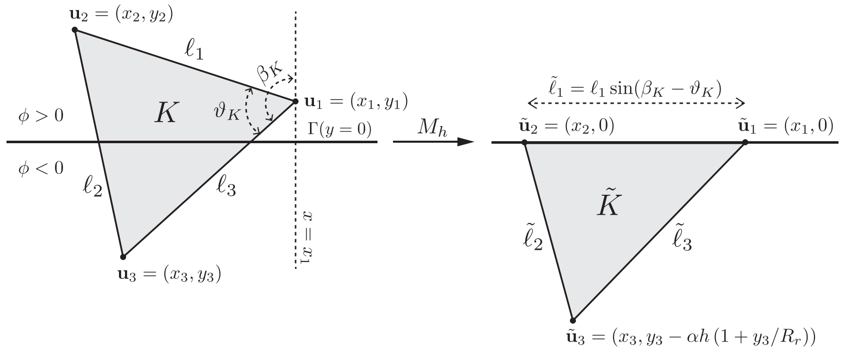
$$x_2 - x_1 \leqslant 0 \quad \text{and} \quad x_3 - x_1 \leqslant 0. \tag{24}$$

Since $\sin \vartheta_K > 0$, we have

$$\sin \vartheta_K = |\sin \vartheta_K| = \sqrt{1 - \cos^2 \vartheta_K} = \frac{(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)}{\ell_1 \ell_3}. \tag{25}$$

Introduce angle β_K defined by the relations

$$\cos \beta_K := \frac{y_3 - y_1}{\ell_3} \quad \text{and} \quad \sin \beta_K := \frac{x_1 - x_3}{\ell_3}. \tag{26}$$



$\mathbf{u}_2 = (x_2, y_2)$
ℓ_1
β_K
$\mathbf{u}_1 = (x_1, y_1)$
$\phi > 0$
K
ϑ_K
$\Gamma(y = 0)$
M_h
$\phi < 0$
ℓ_2
ℓ_3
$x = x_1$
$\mathbf{u}_3 = (x_3, y_3)$
$\tilde{\ell}_1 = \ell_1 \sin(\beta_K - \vartheta_K)$
$\tilde{\mathbf{u}}_2 = (x_2, 0)$
$\tilde{\mathbf{u}}_1 = (x_1, 0)$
$\tilde{K}$
$\tilde{\ell}_2$
$\tilde{\ell}_3$
$\tilde{\mathbf{u}}_3 = (x_3, y_3 - \alpha h\,(1 + y_3/R_r))$

Figure 4. Illustration explaining the rationale behind the acute conditioning angle assumption. Triangle K shown in the figure is positively cut by Γ, which is depicted as being (locally) straight. By projecting its vertices u_1 and u_2 onto Γ, the length of its positive edge is scaled by a factor of $\sin(\beta_K - \vartheta_K)$, where ϑ_K is the conditioning angle of K and β_K is defined in the figure (or by (26)). Since $y_1 \geqslant 0$ and $y_3 < 0$ together imply $\beta_K > 90°$, requesting that ϑ_K be strictly acute ensures that the lengths of positive edges in the conforming mesh determined by the meshing algorithm are bounded away from zero. This in turn helps guarantee that the images of positively cut triangles have good quality.

By direct substitution, we obtain

$$\cos(\beta_K - \vartheta_K) = \cos\beta_K \cos\vartheta_K + \sin\beta_K \sin\vartheta_K = \frac{y_2 - y_1}{\ell_1}. \tag{27}$$

which in turn shows that

$$\tilde{\ell}_1 = |x_1 - x_2| = x_1 - x_2 = \ell_1 |\sin(\beta_K - \vartheta_K)|. \tag{28}$$

Using (22) and (24) in the definition of β_K implies $\cos\beta_K < 0$ and $\sin\beta_K > 0$. Hence, $90° < \beta_K < 180° \Rightarrow 90° - \vartheta_K < \beta_K - \vartheta_K < 180° - \vartheta_K$. Therefore, (28) in fact shows that

$$\cos\vartheta_K \leqslant \frac{\tilde{\ell}_1}{\ell_1} = \sin(\beta_K - \vartheta_K) \leqslant 1. \tag{29}$$

Equation (29) reveals quite clearly, the rationale behind the acute conditioning angle assumption. Without $\vartheta_K < 90°$, x_1 can be arbitrarily close to x_2 or even equal to it. Then $\tilde{\ell}_1$ can be arbitrarily small, resulting in $\tilde{K}$ becoming poorly shaped or degenerate. Equation (29) also helps identify a simple strategy to improve the qualities of positive edges, i.e., ones in which $\tilde{\ell}_1/\ell$ is not too small, particularly when conditioning angles are close to 90°. The idea is described in detail in Section 3.5 and consists in perturbing vertices to 'eliminate' positively cut triangles having β_K close to 90°. Equation (26) shows that this happens when $y_1 - y_3 = \phi(u_1) - \phi(u_3)$ is small compared to the triangle size. In such triangles, we simply perturb u_3 to $\pi(u_3)$; this operation alters the set of positively cut triangles (K is then no longer positively cut) to ensure that positive edges have better quality.

Equation (29) also helps estimate the area of triangle $\tilde{K}$:

$$\Delta_{\tilde{K}} = \frac{1}{2}\begin{vmatrix} 1 & x_1 & 0 \\ 1 & x_2 & 0 \\ 1 & x_3 & y_3 - \alpha h\left(1 + \frac{y_3}{R_r}\right) \end{vmatrix} = \frac{1}{2}(x_1 - x_2)\left(\alpha h - \mu y_3\right). \tag{30}$$

Using (29) and the (trivial) bound $y_3 \geqslant -\ell_3$ in (30) yields

$$\frac{1}{2}\alpha h \ell_1 \cos\vartheta_K \leqslant \Delta_{\tilde{K}} \leqslant \frac{1}{2}\ell_1(\alpha h + \mu\ell_3). \tag{31}$$

To compute the quality $Q(\tilde{K})$ of $\tilde{K}$, we estimate the lengths $\tilde{\ell}_2$ and $\tilde{\ell}_3$ as follows. We have

$$\tilde{\ell}_2^2 = (x_2 - x_3)^2 + \left(y_3 - \alpha h\left(1 + \frac{y_3}{R_r}\right)\right)^2 = (x_2 - x_3)^2 + (\mu y_3 - \alpha h)^2. \tag{32}$$

Similarly for $\tilde{\ell}_3$. Analogous to (18), we obtain

$$\alpha^2 h^2 \leqslant \tilde{\ell}_i^2 \leqslant \ell_i^2 + (\alpha h + \mu \ell_i)^2 \quad \text{for } i = 2, 3. \tag{33}$$

A simple bound for the quality of $\tilde{K}$ hence follows as

$$\frac{\min\{\cos\vartheta_K, \alpha \min_i \tau_i\}}{\sqrt{1 + (\alpha \max_i \tau_i + \mu)^2}} \leqslant \frac{Q(\tilde{K})}{Q(K)} \leqslant \frac{\sqrt{1 + (\alpha \max_i \tau_i + \mu)^2}}{\min\{\cos\vartheta_K, \alpha \min_i \tau_i\}}. \tag{34}$$

Examining the bounds for the measure $\Delta_{\tilde{K}}$ of $\tilde{K}$ in (9), (14), and (31) suggests how to choose parameters α and R_r. Specifically, we would like to ensure that μ is as close to 1 as possible, while making α as small as possible. Hence we select $\alpha = (1 + h/R_r)^{-1}$ and $R_r \gg h$, the former being a consequence of the requirement identified in Section 2.5.3. In the example discussed in Section 2.3 (and Figure 2), the value $\alpha = 0.75$ was in fact chosen this way, see also Table III.

Among the triangles in $\mathcal{T}_h^{\text{sub}}$ that are perturbed by M_h, it remains to examine ones that have either one or two vertices in $\Omega \cap B(\Gamma, R_r)$, and the remaining vertices in $\Omega \setminus B(\Gamma, R_r)$. Vertices of these triangles that lie in $\Omega \cap B(\Gamma, R_r)$ are relaxed according to (3) while the ones in $\Omega \setminus B(\Gamma, R_r)$ remain unperturbed. We omit a discussion of these two cases for brevity sake and only note that

Table III. Qualities of meshes produced by the meshing algorithm with parameters $R_r = 3h$ and $\alpha = 3/4$ for a circular domain of radius R immersed in background meshes of equilateral triangles.The second, third, and fourth columns list in order, the minimum angle, maximum angle, and the maximum ratio of the circumradius to the inradius among triangles in the conforming mesh (similar to Table II). The metric Q introduced in the text is the ratio of the longest and shortest edge lengths in a triangle. The values under columns titled Q_{n-} for $n = 1, 2, 3$ report the maximum values of Q over triangles with n vertices inside the domain and the remaining $3-n$ vertices on the boundary. Compare these values with the estimates for Q in (36) that corresponding to the limit $h/R \to 0$.

h/R	$\theta_{\min}$	$\theta_{\max}$	q	Q_{3-}	Q_{2-}	Q_{1-}
1/12	34.5°	100.1°	2.85	1.27	1.67	1.74
1/24	28.8°	108.1°	3.38	1.27	1.73	1.98
1/48	25.5°	115.5°	4.02	1.28	1.71	2.10
1/96	24.9°	115.0°	4.01	1.28	1.75	2.15

guaranteeing their quality essentially requires that R_r be sufficiently large. The critical cases of transformed triangles in $\mathcal{T}_h^{\text{sub}}$ are the ones we have considered earlier.

The quality estimates in (12), (21), and (34) apply also to adaptively refined meshes. Since the parameter τ_i can be quite large in such meshes, some triangles in $\mathcal{T}_h^c$ can become poorly shaped (but still remain non-degenerate as we have demonstrated with bounds for their areas). Alternative choices for f_h can certainly help improve the mesh quality when triangles near the boundary in the background mesh have widely varying sizes.

2.5.5. Evaluating quality estimates. Let R_r equal three times the mesh size. In Section 2.5.3, we identified the requirement

$$\alpha \geqslant \left(1 + \frac{h}{R_r}\right)^{-1} = \frac{3}{4}, \tag{35}$$

and hence $\mu = 1 - \alpha h/R_r = 3/4$. In a background mesh of equilateral triangles, $\tau_i = 1$ for $i = 1, 2, 3$. For these specific parameters, the upper bounds for Q estimated in (12), (21), and (34) reduce to

$$Q \leqslant \begin{cases} 4/3 & \text{for triangles with three vertices inside the domain,} \\ 2\sqrt{13}/3 \text{ (approx. 2.4)} & \text{for triangles with exactly two vertices inside the domain,} \\ \sqrt{13} \text{ (approx. 3.6)} & \text{for triangles with exactly one vertex inside the domain.} \end{cases} \tag{36}$$

We use the example of a circular domain of radius R immersed in background meshes of equilateral triangles to scrutinize the bounds for Q in (36). Table III reports the qualities of meshes conforming to the circle as the background mesh is refined. The most refined background mesh in the table has mesh size that is about 1/100-th of the radius of the circle. In the resulting conforming mesh, the maximum value of Q over triangles with all three vertices inside the domain, with exactly two vertices in the domain, and with exactly one vertex in the domain are 1.28, 1.75, and 2.15, respectively. These values compare reasonably well with corresponding ones 1.3, 2.4, and 3.6 in (36) (albeit for the case $h/R \to 0$).

We conclude this section mentioning that the calculations in Sections 2.5.2–2.5.4 show that when $h/r_n \ll 1$, the mapping from $\mathcal{T}_h^{\text{sub}}$ to $\mathcal{T}_h^c$ is invertible triangle-wise. Guaranteeing success of the meshing algorithm, even with $h/r_n \ll 1$, additionally requires showing that this transformation is *globally* invertible which we have not included here. It entails demonstrating that distinct triangles in $\mathcal{T}_h^c$ do not overlap.

The general case where h/r_n is small but finite necessarily requires more detailed calculations. An important step to this end is showing that $\pi : \Gamma_h \to \Gamma$ is a homeomorphism onto Γ. For then, we know that positive edges are mapped to non-degenerate and non-overlapping ones that interpolate the boundary. We proved this result in [41] where we also computed bounds for the Jacobian of this mapping and estimated the mesh size required near the boundary. These calculations serve as a point of departure to guarantee the success of the meshing algorithm and to identify the requisite restrictions on the mesh size.

3. REMARKS ON IMPLEMENTATION

We discuss a few details relevant to practical implementations of the meshing algorithm. Included also is a simple vertex perturbation step that alters the collection of positively cut triangles and positive edges to improve the quality of triangles along the boundary in the conforming mesh.

3.1. *Identifying vertices in Ω*

Identifying positive edges in the background mesh requires examining which vertices lie inside the domain and which ones lie outside, i.e., computing $\mathfrak{s}$ at the vertices. This is simplest when Ω is represented implicitly, say as $\Omega := \{x \in \mathbb{R}^2 : \Psi(x) < 0\}$. For then, a vertex v lies in Ω if and only if $\Psi(v) < 0$. When such a level set function Ψ is not known *a priori*, we choose it to be the signed distance function ϕ itself. Note that it suffices to compute $\mathfrak{s}$ (and hence ϕ) at vertices that are close to the boundary— just the ones in its R_r-neighborhood. Such vertices can be conveniently identified using local bounding boxes around the boundary.

3.2. *Closest point projection*

Mapping vertices of positive edges onto Γ requires computing the closest point projection π there. For C^2-regular domains, [38, Theorem 1.5] shows that π and ϕ at a point x sufficiently close to the boundary, namely in $B(\Gamma, r_n)$, are related as

$$\pi(x) = x - \phi(x)\nabla\phi(x). \tag{37}$$

Once ϕ and its gradient are known, (37) then shows how to compute π. Since positive edges are by definition within a distance h from the boundary, relation (37) can be used to evaluate π if the background mesh is sufficiently refined close to the boundary. We refer to [39, Appendix A] for a discussion on computing ϕ and π when Γ is represented either implicitly or parametrically.

3.3. *Tolerances and round-off*

While identifying which vertices lie inside the domain and which ones lie outside, the effect of tolerances and round-off is perhaps unavoidable. As a result, a vertex in Ω may be (mis)identified as one in $\mathbb{R}^2 \setminus \Omega$ and vice versa. The effect of tolerances can be understood as introducing small perturbations in the boundary. Incorrectly identifying vertices in Ω will change the collection of positively cut triangles, positive edges, and hence the resulting mesh for Ω. However, the resulting mesh will be valid provided the assumptions discussed in Section 2.5 are satisfied. In particular, if triangles in the vicinity of the Γ are all acute-angled, the choice of tolerances and the effect of round-off errors is not critical. The resulting mesh may depend on their choices but will be valid nonetheless.

3.4. *Background meshes*

The polygon triangulated by the background mesh is quite arbitrary; it only needs to contain $\overline{\Omega}$. Bearing this in mind, the restrictions on the background mesh discussed in Section 2.5, namely that its triangles near Γ be sufficiently refined and that specific angles be acute, are hence easy to satisfy. A simple way to satisfy the latter is to ensure that triangles in the vicinity of Γ are all acute-angled. Or even adopt a background mesh of all acute-angled triangles. For instance, use a mesh of all equilateral triangles as done in the example in Figure 2.

In practice, it is desirable that the background mesh be adaptively refined depending on the geometric features of the boundary and the solution being approximated. Adaptively refined quadtrees are a convenient way to construct such meshes. Angles in stencils designed by Bern *et al.* in [42] lie between 36° and 80°. Hence, the resulting background meshes automatically satisfy

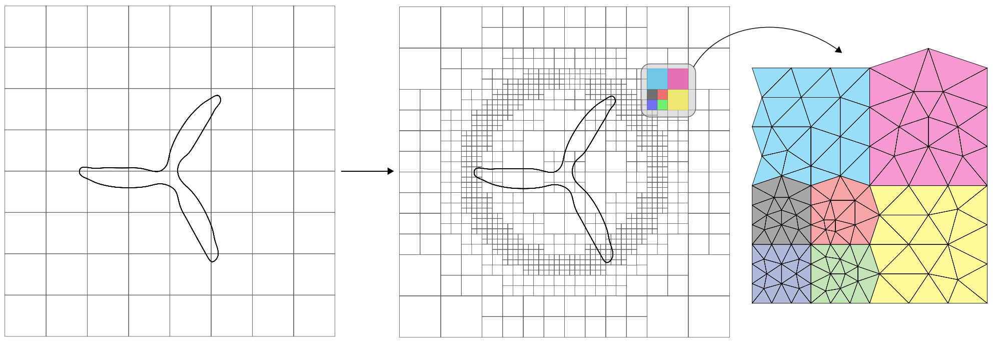


Figure 5. Example showing the construction of an adaptively refined background mesh of acute-angled triangles using quadtrees. In Section 6, we consider the problem of a flow driven by a propeller. Simulating the problem requires a background mesh that is refined along the trajectory of the propeller tips. Following [42], we construct such a mesh by creating an adaptively refined quadtree. Cells in such a tree are then tiled with stencils of strictly acute-angled triangles; the angle requirements in the meshing algorithm are hence automatically satisfied. The resulting mesh is shown in Figure 12.

the angle requirements for the success of the meshing algorithm. The background mesh shown in Figure 12 was constructed in this way, and the steps involved are depicted in Figure 5; more examples of background meshes constructed from adaptively refined quadtrees can be found in [39].

3.5. Altering the collection of positively cut triangles

When the background mesh $\mathcal{T}_h$ is sufficiently refined near Γ, triangles with all three vertices in Ω undergo only small perturbations and hence their quality is not expected to deteriorate significantly. Triangles with exactly one or exactly two vertices in Ω can however be subject to large perturbations. In Sections 2.5.3 and 2.5.4, we showed that these triangles remain non-degenerate and have bounded quality for the special case $h/r_n \to 0$; similar calculations will show that these conclusions hold for small but finite values of h/r_n as well. It is still possible that the qualities of these triangles are not as good as desired. In the following, we briefly describe a simple, heuristic vertex perturbation strategy that (possibly) alters the collection of positively cut triangles to improve the quality of their images in the conforming mesh.

Consider a positively cut triangle K in $\mathcal{T}_h$ with vertices $\{a, b, c\}$ ordered such that $\phi(a) \geqslant \phi(b) \geqslant \phi(c)$. Hence, the edge joining a and b is the positive edge of K and b is its proximal vertex. Let η be a parameter chosen such that $0 \leqslant \eta \ll 1$. The suggested perturbation consists in relocating c to $\pi(c)$ if $\phi(b) - \phi(c) < \eta h_K$. When c is snapped to its closest point on the boundary, K is no longer positively cut. It is necessary to apply these perturbations iteratively because each such perturbation alters the set of positively cut triangles.

The rationale behind the perturbation step is illustrated with an example in Figure 6. In the figure, triangles $K_3 - K_6$ are positively cut by the boundary Γ, which for simplicity has been depicted as being locally straight. Notice that the positive edges in triangles K_3 and K_5 are mapped to relatively short edges. As a result, the qualities of the images $\hat{K}_3$ and $\hat{K}_5$ of K_3 and K_5 respectively, although bounded, could be unsatisfactory.

When vertex v_6 is snapped to its closest point on the boundary, the collection of positively cut triangles is altered. Triangles K_8 and K_{10} are now positively cut, while K_3 and K_5 are not. In fact, K_3 and K_5 are no longer included in the conforming mesh. Table IIb shows the improvement in the quality of meshes for the domain in Figure 2 when these suggested perturbations are included. In many of the examples we have tried, we found that such perturbations yield meshes with better quality triangles along the boundary. A detailed analysis is however required to determine whether we can guarantee such improvement, to identify conditions on η required to this end, and to estimate the number of iterations required.

We conclude this discussion mentioning that mesh smoothing algorithms can be used to further improve the mesh quality [44, 45]. Figure 7 demonstrates how mesh smoothing improves the quali-

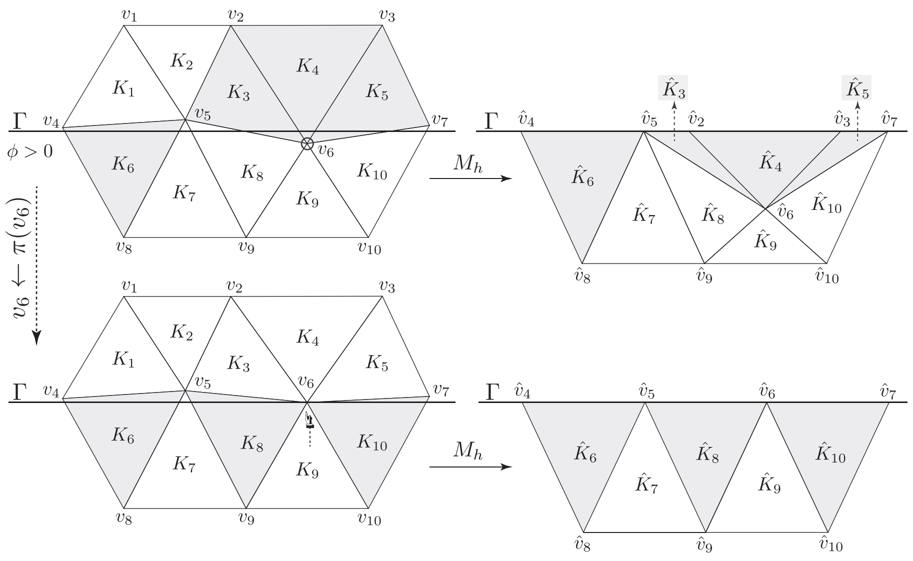


Figure 6. By selectively perturbing vertices in the background mesh, we alter the collection of positively cut triangles to improve the quality of triangles produced by the meshing algorithm. The example here shows that positively cut triangles can become sliver shaped when both vertices of positive edges project to nearby points on the boundary. The acute conditioning angle assumption then implies that in such triangles, the proximal vertex and the vertex inside the domain are both close to Γ. Triangles $\hat{K}_3$ and $\hat{K}_5$ in the mesh on the top right are sliver shaped; the criterion $\phi(v_5) - \phi(v_6) \ll h_{K_3}$ and $\phi(v_7) - \phi(v_6) \ll h_{K_5}$, respectively, helps identify these cases. When v_6 is snapped to its closest point on Γ, K_3 and K_5 are no longer positively cut and the quality of the resulting mesh shown at the bottom right is improved.

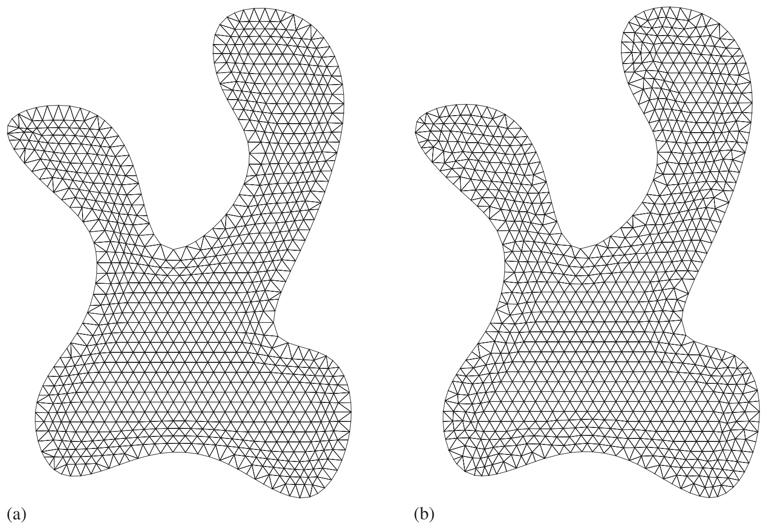


Figure 7. Example demonstrating the idea of using mesh smoothing to further improve the quality of triangulations resulting from our meshing algorithm. Figure (a) shows the triangulation determined by the meshing algorithm for the example discussed in Section 2.3 using the coarsest background mesh $\mathcal{T}_h$ mentioned there and in Tables I and IIa. Its quality is improved in Figure (b) using a mesh smoothing algorithm [43]. To provide a fair comparison, we designed the smoothing operations to be such that they adjust precisely the same set of vertices that were perturbed in the meshing algorithm. Table (c) summarizes the improvement in quality metrics that results from smoothing the meshes determined by the meshing algorithm. We have not used such smoothing in any of the numerical examples presented subsequently.

ties of meshes in the example discussed previously in Section 2.3. Specific details of the smoothing algorithm can be found in [43]. The purpose of the example in Figure 7 is only to convey that our meshing algorithm determines a mesh of reasonable quality that serves as a point of departure for mesh smoothing methods, irrespective of the specific one adopted. We have not used any smoothing in any of the examples that follow.

4. CURVILINEAR TRIANGULATIONS

Curvilinear discretizations provide high-order accurate approximations for curved domains, compared to the polygonal ones that result from the meshing algorithm in Section 2. Curved finite elements constructed using such discretizations are indispensable for optimal accuracy with high-order interpolations. In this section, we are concerned with constructing curvilinear triangulations for C^2-regular domains by transforming straight-edged triangles of a nonconforming background mesh. Constructing such mappings from straight to curved triangles, even with a conforming mesh, is a delicate task because there are two conflicting requirements. The resulting curved triangle should approximate the domain well. Yet, it should be a sufficiently small perturbation of the straight one if interpolation error estimates on the latter are expected to translate into optimal ones over the curved triangle [46]. In the following, we discuss such mappings that generalize the ones in [47, 48].

Curvilinear triangulations broadly fall into two categories. In the first kind, curved triangles conform exactly to the curved boundary. In the other, curved triangles approximate the domain sufficiently well and are usually defined via isoparametric mappings. We discuss the former in Section 4.1 and the latter in Section 4.2. In both constructions, the positive edge in each positively cut triangle is transformed into a curved one to conform/approximate the boundary; rest of the edges remain straight.

4.1. Exactly conforming curved triangles

Defining a curvilinear mesh that conforms exactly to Γ requires only a subtle modification of the meshing algorithm—instead of mapping vertices of positive edges onto Γ, we map positive edges themselves onto Γ. Consider a positively cut triangle K with vertices $\{u, v, w\}$ ordered such that $\mathfrak{s}(u) \geqslant \mathfrak{s}(v) \geqslant \mathfrak{s}(w)$. Denote the barycentric coordinates of $x \in \overline{K}$ by $\{\lambda_u, \lambda_v, \lambda_w\}$ so that $x = \lambda_u\, u + \lambda_v\, v + \lambda_w\, w$ and $\lambda_u + \lambda_v + \lambda_w = 1$. The mapping that transforms K into a curvilinear triangle conforming to Ω is given by

$$\begin{aligned} G_h^K(x) := &\frac{1}{2(1-\lambda_u)}\left[\lambda_v \pi\left(\lambda_u\, u + (1-\lambda_u)\, v\right) + \lambda_u \lambda_w \pi(u)\right] \\ &+ \frac{1}{2(1-\lambda_v)}\left[\lambda_u \pi\left((1-\lambda_v)\, u + \lambda_v\, v\right) + \lambda_v \lambda_w \pi(v)\right] \\ &+ \lambda_w\, M_h(w). \end{aligned} \tag{38}$$

The dependence on x in (38) is implicit in the barycentric coordinates λ_u, λ_v, and λ_w. Over the remaining triangles in $\mathcal{T}_h^{\text{sub}}$, G_h^K is defined just as in the meshing algorithm, as $x \mapsto \lambda_u\, M_h(u) + \lambda_v\, M_h(v) + \lambda_w\, M_h(w)$.

Let us examine the definition of G_h^K for a positively cut triangle K, see Figure 8. By the assumed ordering for vertices, the edge $\overline{uv}$ joining vertices u and v is the positive edge of K. On this edge, $\lambda_w = 0$ and $\lambda_u + \lambda_v = 1$. So

$$G_h^K(x \in \overline{uv}) = \frac{1}{2}\pi((1-\lambda_v)\, u + \lambda_v\, v) + \frac{1}{2}\pi(\lambda_u\, u + (1-\lambda_u)\, v) = \pi(x). \tag{39}$$

Hence G_h^K equals the closest point projection over the positive edge $\overline{uv}$, showing that G_h^K maps positive edges onto Γ as depicted in Figure 8. On the edge $\overline{uw}$, $\lambda_v = 0$ and $\lambda_u + \lambda_w = 1$. Then (38) reduces to

$$G_h^K(x \in \overline{uw}) = \frac{\lambda_u \lambda_w}{2(1-\lambda_u)}\pi(u) + \frac{1}{2}\pi(u) + \lambda_w\, M_h(w) = \lambda_u\, \pi(u) + \lambda_w\, M_h(w), \tag{40}$$

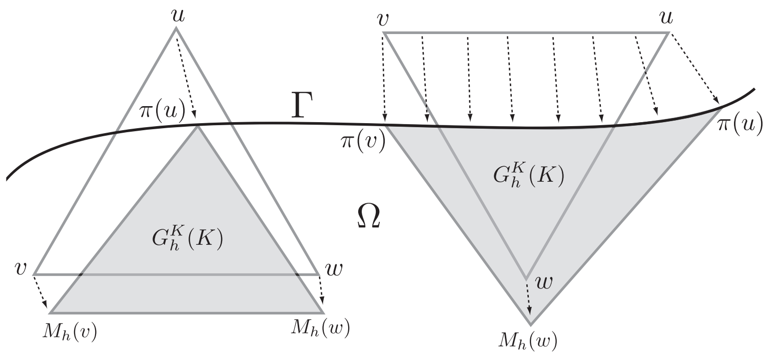


Figure 8. Mapping triangles from a background mesh into curvilinear ones that conform exactly to an immersed boundary. The mapping G_h^K transforms positively cut triangles into curved ones (as shown on the right), while the remaining triangles in $\mathcal{T}_h^{\text{sub}}$ are transformed affinely (as shown on the left).

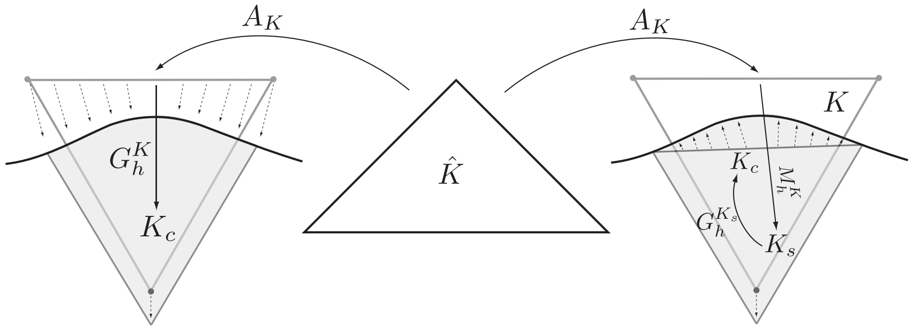


Figure 9. Defining high-order curved finite elements by constructing mappings from the reference element $\hat{K}$ to the curved one K_c. The figure shows two such mappings. The one on the left maps $\hat{K}$ to K_c via the steps $\hat{K} \xrightarrow{A_K} K \xrightarrow{G_h^K} K_c$, while the second one on the right uses the meshing algorithm as an intermediate step, following the sequence $\hat{K} \xrightarrow{A_K} K \xrightarrow{M_h^K} K_s \xrightarrow{G_h^{K_s}} K_c$.

which is an affine map. Similarly,

$$G_h^K(x \in \overline{vw}) = \lambda_v\, \pi(v) + \lambda_w\, M_h(w). \tag{41}$$

Equations (39), (40), and (41) show that G_h^K for a positively cut triangle K can be interpreted as the extension to K of a map that equals π on its positive edge and that is affine on the remaining two edges. This point of view is adopted in [47, 49] as well. For this reason, mappings such as G_h^K are also commonly termed as blending maps and transfinite interpolations.

An alternative way of mapping positively cut triangles to curved ones uses the meshing algorithm as an intermediate step. In such a construction, the domain Ω is first meshed using the algorithm in Section 2, and the resulting mesh then transformed into one that conforms to Γ. More precisely, with

$$M_h^K(x) := \lambda_u(x)\, M_h(u) + \lambda_v(x)\, M_h(v) + \lambda_w(x)\, M_h(w),$$

and $K_s = M_h^K(K)$, the mapping over triangles in $\mathcal{T}_h^{\text{sub}}$

$$\tilde{G}_h^K = \begin{cases} G_h^{K_s} \circ M_h^K & \text{if } K \text{ is positively cut,} \\ M_h^K & \text{otherwise} \end{cases} \tag{42}$$

maps triangles in $\mathcal{T}_h^{\text{sub}}$ to a curvilinear mesh that conforms exactly to Ω. Of course, $\tilde{G}_h^K$ differs from G_h^K only for positively cut triangles, see Figure 9. In these triangles,

(i) $\tilde{G}_h^K$ first maps K to a conforming triangle K_s and then transforms K_s to a curved triangle, and
(ii) even though $G_h^K(\overline{K}) = \tilde{G}_h^K(\overline{K})$ as sets in $\mathbb{R}^2$, $\tilde{G}_h^K \neq G_h^K$ in general. The two will however be close in a pointwise sense.

4.2. Isoparametric mappings

Isoparametric mappings provide a convenient way of approximating curved domains with desired accuracy. To facilitate discussing isoparametric mappings, introduce the reference triangle $\hat{K} \subset \mathbb{R}^2$ and the finite element triplet $(\hat{K}, \hat{N}^k, \hat{\mathbb{P}}^k)$. As usual, $\hat{\mathbb{P}}^k$ is the set of polynomials over $\hat{K}$ of degree at most k and $\hat{N}^k = \{\hat{N}_a\}_a$ is the set of shape functions that constitute a basis for $\hat{\mathbb{P}}^k$. Associated with $\hat{N}^k$ are the nodes $\{\hat{z}_a\}_a \in \hat{K}$ which are such that $\hat{N}_a(\hat{z_b}) = \delta_{ab}$. Define the interpolation operator $\hat{\Pi}^k : f \in [C^0(\hat{K})]^2 \to \sum_a f(\hat{z}_a)N_a$. Finally, let $A_K : \hat{K} \to K$ be an affine map from $\hat{K}$ to K and denote $z_a = A_K(\hat{z}_a)$.

A systematic definition of isoparametric mappings results naturally from interpolating $G_h^K \circ A_K$ or $\tilde{G}_h^K \circ A_K$ at selected points. With the notation introduced earlier, the isoparametric map over $\hat{K} \in \mathcal{T}_h^{\text{sub}}$ defined by interpolating the former is given by

$$I_h^K := \hat{\Pi}^k \left(G_h^K \circ A_K\right) = \sum_a G_h^K(z_a)\hat{N}_a. \tag{43}$$

Interpolating $\tilde{G}_h^K \circ A_K$ instead yields a different map

$$\tilde{I}_h^K := \hat{\Pi}^k \left(\tilde{G}_h^K \circ A_K\right) = \sum_a \tilde{G}_h^{K_s}\left(M_h^K(z_a)\right)\hat{N}_a. \tag{44}$$

For triangles in $\mathcal{T}_h^{\text{sub}}$ that are not positively cut, I_h^K, $\tilde{I}_h^K$, $G_h^K \circ A_K$, and $\tilde{G}_h^K \circ A_K$ are all equal (and affine). In positively cut triangles, I_h^K and $\tilde{I}_h^K$ map positive edges to curved ones that interpolate the boundary at a few points. Figure 10 depicts this for case of a quadratic element ($k = 2$). It also shows that $I_h^K \neq \tilde{I}_h^K$ in general. For instance, $I_h^K(\hat{z}_4) = \pi((z_1 + z_2)/2)$ while $\tilde{I}_h^K(\hat{z}_4) = \pi((\pi(z_1) + \pi(z_2))/2)$. Nonetheless, the two maps will be close for small mesh sizes.

We highlight a few differences between the mapping to curved elements in Section 4.1 and the isoparametric mappings defined here. As pointed out previously, the curved boundary resulting from the former conform to Γ exactly while the curved boundary defined by isoparametric maps can only be expected to interpolate Γ. This could be significant in problems/numerical schemes where retaining the smoothness of the boundary is critical, see Section 5.2 for an example. Isoparametric maps are of course polynomials over coordinates in $\hat{K}$ (or K). As defined in (42) and (43), they require fewer evaluations of π in general. Once π is computed at the vertices of the positive edge, defining I_h^K or $\tilde{I}_h^K$ requires evaluating π at most *twice per node* in the element. In contrast, computing G_h^K or $\hat{G}_h^K$ requires *two evaluations per quadrature point* in the conforming curved element. Furthermore, computing derivatives of shape functions in the isoparametric element do not entail computing

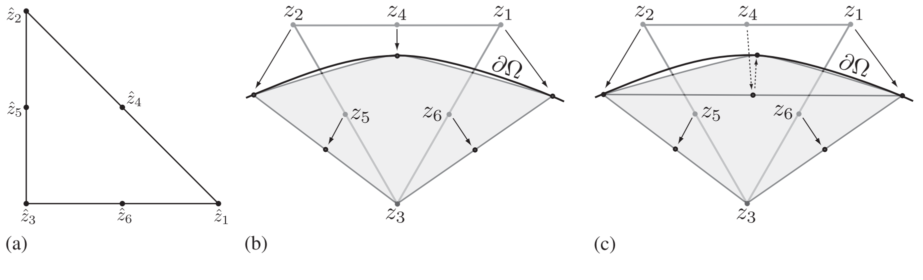


Figure 10. Isoparametric mappings for positively cut triangles in the background mesh. The reference quadratic element is shown in (a). The isoparametric maps constructed by interpolating the mappings $G_h^K \circ A_K$ and $\tilde{G}_h^K \circ A_K$ are shown in (a) and (b). The curved boundary of the element interpolates the boundary at the points where the nodes $\hat{z}_1, \hat{z}_2$, and $\hat{z}_4$ are mapped. Notice that the mid-side node $\hat{z}_4$ is generally mapped differently by G_h^K and $\tilde{G}_h^K$.

derivatives of π; in the conforming curved element however, derivatives of shape functions depend on ∇G_h^K $\left(\text{or } \nabla \tilde{G}_h^K\right)$, which in turn depend on $\nabla \pi$.

5. APPLICATION: HIGH-ORDER CURVED FINITE ELEMENTS

The mappings from straight to curvilinear triangles facilitate a natural construction of curved Lagrange finite elements. The idea behind the construction is illustrated in Figure 9. Following the notation introduced earlier, the curved finite element corresponding to K derived from the reference triplet $(\hat{K}, \hat{N}^k, \hat{\mathbb{P}}^k)$ is denoted by (K_c, N^k, P^k). The curved triangle K_c can be defined using any one of the maps $G_h^K, \tilde{G}_h^k, I_h^k$ or $\tilde{I}_h^k$; for definiteness, we pick the first one. Hence, $K_c = G_h^K(K) = G_h^K(A_K(\hat{K}))$ and

$$P^k = \left\{\hat{p} \circ A_K^{-1} \circ \left(G_h^K\right)^{-1} \; : \; \hat{p} \in \hat{\mathbb{P}}^k\right\}. \tag{45}$$

In particular, shape functions $\{N_a\}_a$ over K_c are defined by the relation $N_a \circ G_h^K \circ A_K = \hat{N}_a$. Nodes $\{z_a^c\}_a$ in the curved element are located at $z_a^c = G_h^K(A_K(\hat{z}_a))$. Hereafter, the set $\{\hat{z}_a\}_a$ will be chosen such that the finite element functions over Ω are in $C^0(\Omega)$.

We demonstrate optimal convergence using the curved finite elements described above with a numerical example. Although the given construction for curved elements is a standard one, the example helps show that the mapping G_h^K in (38) satisfies the conditions in [46] for optimal interpolation estimates. For brevity sake, we do not include similar examples using the maps $\tilde{G}_h^K$, I_h^K, or $\tilde{I}_h^K$ instead of G_h^K but mention that optimal convergence rates would be observed with these maps as well.

We consider the model problem

$$\Delta u = 0 \text{ in } \Omega = \{r = \sqrt{x^2 + y^2} < 1\}, \tag{46a}$$

$$u = e^y \sin x \text{ on } \partial\Omega. \tag{46b}$$

The solution to (46) is the smooth function $u(x, y) = e^y \sin x$. The weak form of (46) is to find $u \in H_\partial^1 := \{v \in H^1(\Omega) : v\big|_{\partial\Omega} = e^y \sin x\}$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, d\Omega = 0 \quad \forall v \in H_0^1(\Omega). \tag{47}$$

To compute finite element approximations u_h of u, Ω is immersed in background meshes of equilateral triangles. The coarsest background mesh has mesh size $h_0 = 5/16$. Table IV shows the convergence of the solution computed with standard Lagrange elements (over $\hat{K}$), as the background mesh is refined (h-refinement). The convergence is illustrated graphically in Figure 11. Dirichlet boundary conditions were imposed by interpolating the prescribed function in (46b) at the nodes of curved elements lying on the boundary. We used sufficiently accurate quadrature rules to evaluate the stiffness matrix, see Section 5.1. The convergence rates in the $L^2(\Omega)$ and $H^1(\Omega)$ norms are optimal for linear, quadratic, cubic, and quartic elements ($k = 1, 2, 3$, and 4, respectively).

Table 11 also reveals that for a given background mesh, the error in the finite element solution decreases with the element order k. This demonstrates that the curved elements are well suited for p-refinement—progressively accurate solutions can be computed by just increasing the element order while using the *same background mesh*. Moreover, the data in the table show that the error is $\mathcal{O}(h^{k+1})$, which is optimal in the element order k for each given (sufficiently small) mesh size h of the background mesh.

Table IV. Convergence of the error of the finite element solutions u_h computed using exactly conforming curved elements for problem (46). Table IVa reports $\|u - u_h\|_0$, the $L_2(\Omega)$-norm of the error $e_h = u - u_h$. The *rate* of convergence with mesh size is defined as the exponent m in expression $\|e_h\|_0 = Ch^m$, where the constant C is independent of h. We calculate a rate for each pair of solutions computed with a background mesh and its refinement; hence the rate calculated from the solutions computed using background meshes $\mathcal{T}_h$ and $\mathcal{T}_{h/2}$ is $(\log_2 \|e_h\|_0 - \log_2 \|e_{h/2}\|_0)$. Table IVb reports analogous optimal convergence of the error in the $H^1(\Omega)$-seminorm. We see that convergence rates for linear, quadratic, cubic, and quartic elements are optimal both in the $L^2(\Omega)$ and $H^1(\Omega)$ norms. Figure 11 presents the data in these tables in graphical format.

(a) Convergence of the $L^2(\Omega)$-norm of the error

mesh size	Linear (P1)		Quadratic (P2)		Cubic (P3)		Quartic (P4)	
	error	rate	error	rate	error	rate	error	rate
h	1.053×10^{-2}	–	4.059×10^{-4}	–	2.654×10^{-5}	–	1.716×10^{-6}	–
$h/2$	2.854×10^{-3}	1.88	4.342×10^{-5}	3.22	1.457×10^{-6}	4.19	3.567×10^{-8}	5.59
$h/4$	7.705×10^{-4}	1.89	5.473×10^{-6}	2.99	1.046×10^{-7}	3.80	1.161×10^{-9}	4.94
$h/8$	1.941×10^{-4}	1.99	6.594×10^{-7}	3.05	6.838×10^{-9}	3.94	2.690×10^{-11}	5.43
$h/16$	4.862×10^{-5}	2.00	7.965×10^{-8}	3.05	4.279×10^{-10}	4.00	–	–
$h/32$	1.212×10^{-5}	2.00	9.714×10^{-9}	3.04	2.690×10^{-11}	3.99	–	–
$h/64$	3.022×10^{-6}	2.00	–	–	–	–	–	–

(b) Convergence of the $H^1(\Omega)$-seminorm of the error

mesh size	Linear (P1)		Quadratic (P2)		Cubic (P3)		Quartic (P4)	
	error	rate	error	rate	error	rate	error	rate
h	2.639×10^{-1}	–	1.533×10^{-2}	–	9.465×10^{-4}	–	9.400×10^{-5}	–
$h/2$	1.349×10^{-1}	0.97	3.410×10^{-3}	2.17	8.854×10^{-5}	3.42	4.047×10^{-6}	4.53
$h/4$	6.935×10^{-2}	0.96	8.379×10^{-4}	2.03	1.017×10^{-5}	3.12	2.476×10^{-7}	4.03
$h/8$	3.538×10^{-2}	0.97	2.016×10^{-4}	2.05	1.091×10^{-6}	3.22	1.136×10^{-8}	4.45
$h/16$	1.779×10^{-2}	0.99	4.899×10^{-5}	2.04	1.189×10^{-7}	3.19	–	–
$h/32$	8.910×10^{-3}	1.00	1.201×10^{-5}	2.03	–	–	–	–
$h/64$	4.454×10^{-3}	1.00	–	–	–	–	–	–

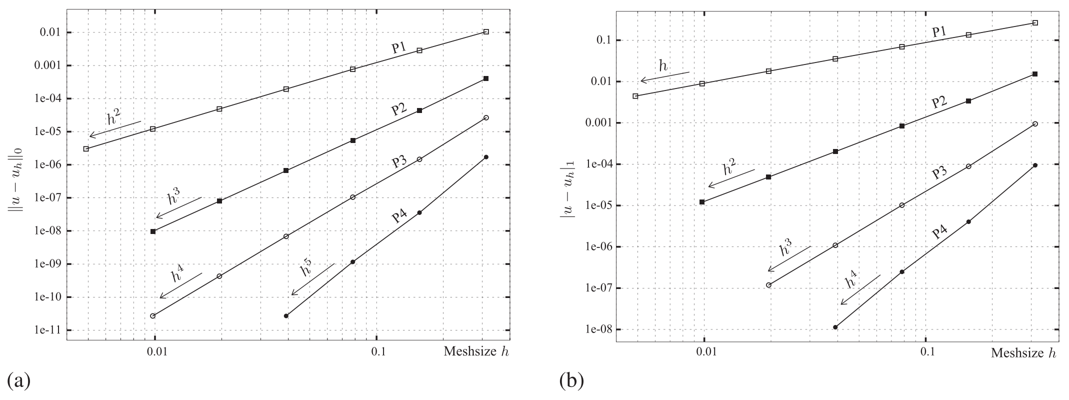


Figure 11. Graphical illustration of the optimal convergence of the finite element solution u_h to the exact one for the problem (46). Figures 11a and 11b show convergence of the $L^2(\Omega)$-norm and $H^1(\Omega)$-seminorm respectively, of the error in the finite element solution as the background mesh is refined. The rates of convergence are optimal in both norms for linear, quadratic, cubic, and quartic elements.

5.1. *Quadrature for curved elements*

For optimal convergence and accuracy of numerical solutions computed using curved elements, we naturally require sufficiently accurate quadrature rules for integration over curvilinear domains. Following standard practice, these quadrature rules need only be defined over the reference triangle, because integrals over a curved element K_c can be performed over $\hat{K}$ using the correspondence provided by the mappings to K_c $\left(\text{any one of } G_h^K, \tilde{G}_h^K, I_h^K, \tilde{I}_h^K\right)$. We adopted for curved elements the same quadrature rules needed for straight elements, as explained in [50]. For example, a three-point quadrature rule that exactly integrates quadratic polynomials over the reference element suffices for isoparametric quadratic elements. We have used such integration rules in all of our examples, including the ones with exactly conforming curved elements. These examples suggest that the quadrature rules for straight elements are also enough to obtain optimal convergence rates with exactly curved elements. We also note that more generally, different rules can be used for integrating each term in the weak form of a problem, for instance the mass and stiffness matrices, and the force vector.

5.2. Circular plate in bending

We consider the problem of a thick, circular, elastic plate bending under the action of a uniform external load from [36]. This example helps emphasize the distinction between representing a curved boundary exactly using exactly conforming curved elements and approximating it with isoparametric elements as defined earlier. Consider a Cartesian coordinate system (x, y, z) with basis $\{e_x, e_y, e_z\}$. The domain of the problem is the set $\Omega \times (-\tau/2, \tau/2)$, where Ω is a circle of radius $R = 3.142$ centered at the origin and contained in the plane of e_x, e_y, and τ is the thickness along e_z. We assume that the displacement field u of the plate is of the form

$$u(x, y, z) = -z\left(\vartheta_x(x, y)e_x + \vartheta_y(x, y)e_y\right) + w(x, y)e_z, \tag{48}$$

which corresponds to the Reissner–Mindlin model for a thick plate in bending, see [36, 51]. In (48), w is the transverse displacement of points in the mid-plane Ω, while ϑ_x and ϑ_y represent the infinitesimal rotations of fibers normal to the mid-plane about the axes e_y and e_x, respectively. We consider a 'soft-simple support' for the plate, which implies the boundary conditions

$$\left.\begin{aligned} w &= 0 \\ \vartheta \cdot T &= 0 \end{aligned}\right\} \text{ on } \partial\Omega \tag{49}$$

where T is the unit tangent to $\partial\Omega$ and $\vartheta = (\vartheta_x, \vartheta_y)$. The plate is loaded by a constant force $2p$ per unit area normal to its top face $\Omega \times \{\tau/2\}$. The elasticity problem is then to find

$$\begin{aligned} u &\in \left\{-z\,\vartheta + w\,e_z \;:\; \vartheta \in H^1_T, w \in H^1_0(\Omega)\right\}, \\ \text{where } \; H^1_T(\Omega) &:= \{\vartheta \in [H^1(\Omega)]^2 \;:\; \vartheta \cdot T = 0 \text{ on } \partial\Omega\}, \end{aligned}$$

which minimizes the strain energy functional

$$I[u] = \frac{1}{2}\int_{\Omega\times(-\tau/2,\tau/2)} \left\{\lambda[\mathrm{tr}(\varepsilon(u))]^2 + 2\mu\,\varepsilon(u) : \varepsilon(u)\right\} - \int_\Omega p\,w, \tag{50}$$

where λ, μ are material parameters called Lamé constants and $\varepsilon(u) = (\nabla u + \nabla u^T)/2$ is the usual infinitesimal strain tensor. Introducing assumption (48) in (50) and integrating along the thickness reduces (50) to a problem over Ω: find $(\vartheta, w) \in H^1_T(\Omega) \times H^1_0(\Omega)$ that minimizes the functional

$$\begin{aligned} F[(\vartheta, w)] = \frac{1}{2}\int_\Omega &\left\{\lambda\,[\mathrm{tr}\,(\varepsilon(\vartheta))]^2 + 2\mu\,\varepsilon(\vartheta) : \varepsilon(\vartheta)\right\} \\ &+ \frac{6\mu}{\tau^2}\int_\Omega \|\vartheta - \nabla w\|^2 - \frac{12}{\tau^3}\int_\Omega pw. \end{aligned} \tag{51}$$

We compare the transverse displacements at the center of the plate as the background mesh for Ω is refined while using curved quadratic elements. We pick $\lambda = \mu = 1, \tau = R/4$ for the plate and $p = 1 \times 10^{-3}$ for the loading. The constraint (49) is imposed at the nodes that lie on $\partial\Omega$. Hence with the notation used in Figure 10, (49) is imposed at nodes z_1, z_2, and z_4 in each curved quadratic element. We take as a reference value $w_0 = 5.3407075 \times 10^{-2}$, computed with exactly conforming curved quartic elements and a refined background mesh of equilateral triangles. Table V lists the

Table V. Transverse displacements at the center of the circular plate Ω computed with curved quadratic elements. The reference value is $w_0 = 5.3407075 \times 10^{-2}$. The columns titled 'conforming' and 'isoparametric' list the values computed with exactly conforming elements and isoparametric elements, respectively. The column 'modified isoparametric' contains the values computed with isoparametric elements but while imposing the constraint $\vartheta \cdot T = 0$ using one of the two possible tangents T at vertices on the curved edges of the element. The coarsest mesh size of the background mesh is $h_0 \simeq 0.28R$.

mesh size	conforming ($\times 10^{-2}$)	isoparametric ($\times 10^{-2}$)	modified isoparametric ($\times 10^{-2}$)
h_0	5.3370662	2.2343001	5.3383666
$h_0/2$	5.3401468	1.9557577	5.3400195
$h_0/4$	5.3406606	1.8019916	5.3406438
$h_0/8$	5.3407108	1.6864489	5.3407156

displacements computed with exactly conforming quadratic elements ($K_c = G_h^K(K), k = 2$) and with isoparametric quadratic elements ($K_c = I_h^K(\hat{K}), k = 2$).

From the table, we see that the displacements computed with the conforming elements converge to w_0. But somewhat surprisingly, those computed with quadratic isoparametric elements fail to even come close to w_0. This is a consequence of enforcing the constraint on rotations in (49) on the approximate curved boundary realized with isoparametric elements. The unit tangent to this boundary fails to be continuous at vertices that lie on it. Consequently, rotations equal zero at each vertex on the boundary. With exactly conforming curved elements, $\partial\Omega$ is represented exactly and this issue is avoided.

The above discussion shows that the constraint in (49) needs to be enforced differently while using isoparametric elements. For instance, we could select one of the tangents at each vertex on the approximate boundary to enforce the constraint $\vartheta \cdot T = 0$. The displacements computed with quadratic isoparametric elements by enforcing the constraint in this way are listed under the column title 'modified isoparametric' in Table V. These values are clearly more accurate and converge to w_0.

We conclude this example mentioning that we have used the same finite element spaces for both transverse displacements (w) and rotations (ϑ_x, ϑ_y) in the aforementioned calculations. It is well known that for thin plates ($\tau \ll R$), such a choice of spaces in the Reissner–Mindlin model results in locking [51]. To avoid adopting very small mesh sizes for accuracy, we deliberately chose a large thickness $\tau = R/4$ in the example.

6. BACKGROUND MESHES AS UNIVERSAL MESHES

An important advantage of admitting nonconforming background meshes is found in problems with evolving domains. For then, it is possible at least in principle, to use the same background mesh to triangulate a changing domain. If not for the entire duration of interest, at least for reasonably large changes in the immersed geometry. This motivates the notion of *universal meshes*.

6.1. Universal meshes

Consider simulating a problem in which rigid blades physically mix fluid in a closed container. As the blades rotate, the region of the container occupied by the fluid changes. With the algorithm in Section 2, we can now discretize the evolving fluid domain by merely perturbing vertices of the *same* background mesh shown in Figure 12. Precisely because the same background mesh is utilized for all positions of the propeller, we term it a *universal mesh* for the fluid.

More generally speaking, given a triangulation $\mathcal{T}_h$, let $\mathcal{D}(\mathcal{T}_h)$ denote the class of all domains that can be meshed with the algorithm in Section 2 using $\mathcal{T}_h$ as a background mesh. We say that $\mathcal{T}_h$ is a *universal mesh* for domains in $\mathcal{D}(\mathcal{T}_h)$. The utility of this concept lies in the fact that if $\{\Omega_t\}_t$ is the time evolution of a domain $\Omega_0 \in \mathcal{D}(\mathcal{T}_h)$, then often $\{\Omega_t : 0 \leqslant t \leqslant \tau\} \subset \mathcal{D}(\mathcal{T}_h)$ for a reasonably large time $\tau > 0$. As the domain develops small features or undergoes topological changes, it may no longer belong to $\mathcal{D}(\mathcal{T}_h)$.

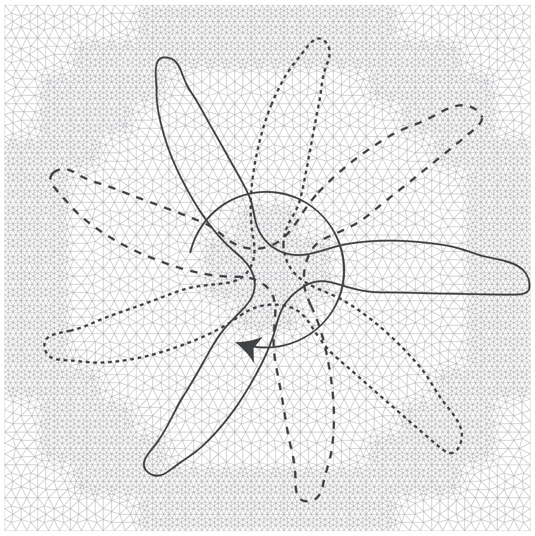


Figure 12. Example to illustrate the notion of a universal mesh. A three-blade propeller rotates about an axis perpendicular to its plane and passing through its center. It is immersed in a refined background mesh of acute-angled triangles. Using this background mesh in the meshing algorithm in Section 2.2 yields a conforming mesh for each orientation of the propeller; a few are shown in Figure 13. Such a background mesh is hence termed a universal mesh for the domain of the propeller.

Figure 12 illustrates this idea for the example with a rotating propeller. The domain shown in the figure is a three-blade propeller that rotates about an axis perpendicular to its plane and passing through the center. The background mesh in the figure consists of only acute-angled triangles. It is refined at the center and along the tips of the blades to resolve the larger curvatures there. This mesh yields a discretization conforming to the fluid for every orientation of the propeller; it is therefore a universal mesh for the fluid. Since the background mesh is quite refined, we show conforming meshes for the propeller in Figure 13 in place of meshes for the fluid. Figure 14 shows the triangles near the propeller boundary in conforming meshes for the fluid for a couple of different orientations of the propeller.

An important question in practice is knowing when Ω belongs to $\mathcal{D}(\mathcal{T}_h)$. Precisely characterizing $\mathcal{D}(\mathcal{T}_h)$ is presumably very difficult. The results in Section 2.5 are useful toward this end. By identifying sufficient conditions for a domain to belong to $\mathcal{D}(\mathcal{T}_h)$, these results enable verifying the inclusion of a certain family of domains in $\mathcal{D}(\mathcal{T}_h)$. Improving on these conditions, by way of relaxing the assumption on angles and tightening the bound on the required mesh size, will help identify a larger family of domains in $\mathcal{D}(\mathcal{T}_h)$.

Next, we present a couple of applications of universal meshes using the curved elements described in Section 4.1. In the first example, we consider the flow of a Stokes fluid through a square-shaped channel having a propeller rotating with fixed angular velocity. In the second, we consider a similar setup in which the fluid interacts with a rigid circular obstacle that is tethered by a spring to the container wall but otherwise free to move within the container. Both examples effectively reduce to computing solutions to the Stokes equations over a sequence of (changing) domains and boundary conditions. These domains are robustly and exactly discretized with the same background mesh that serves as their universal mesh.

6.2. Application: flow with a rotating component

First, we consider the example of a propeller mixing fluid. The problem setup is essentially the one illustrated in Figure 12. The propeller P is assumed to be rigid and impermeable. It rotates with constant angular velocity ω about an axis passing through its center and perpendicular to its plane. The fluid in the container is incompressible and has viscosity μ. Its kinematics is governed by the

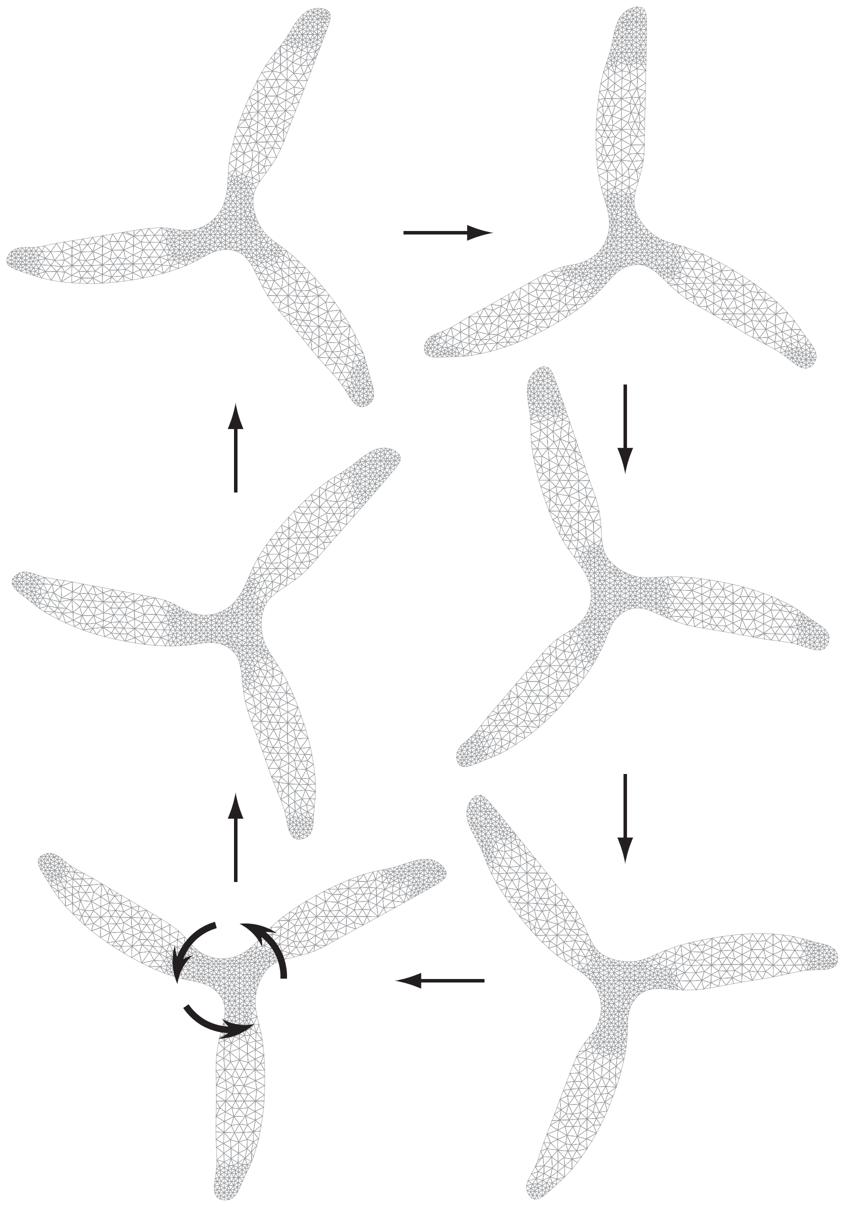


Figure 13. Conforming meshes for a few different orientations of a propeller-shaped domain determined by the meshing algorithm using the same background mesh shown in Figure 12.

familiar equations for Stokes flow,

$$\mu \operatorname{div}(\nabla u_t) = \nabla p_t, \tag{52a}$$

$$\operatorname{div}(u_t) = 0, \tag{52b}$$

relating the flow velocity u_t and pressure p_t at time t. Let $\{e_x, e_y\}$ and $\{e_r, e_\theta\}$ denote Cartesian and polar coordinate systems with origin at the center of the container of size $L = 1.4$. Inflow and outflow boundary conditions along its walls are specified as

$$u_t = \begin{cases} 0 & \text{if } |y| = L/2, \\ (L - 2|y|)\, e_x & \text{if } x = 0, L \end{cases} \tag{53}$$

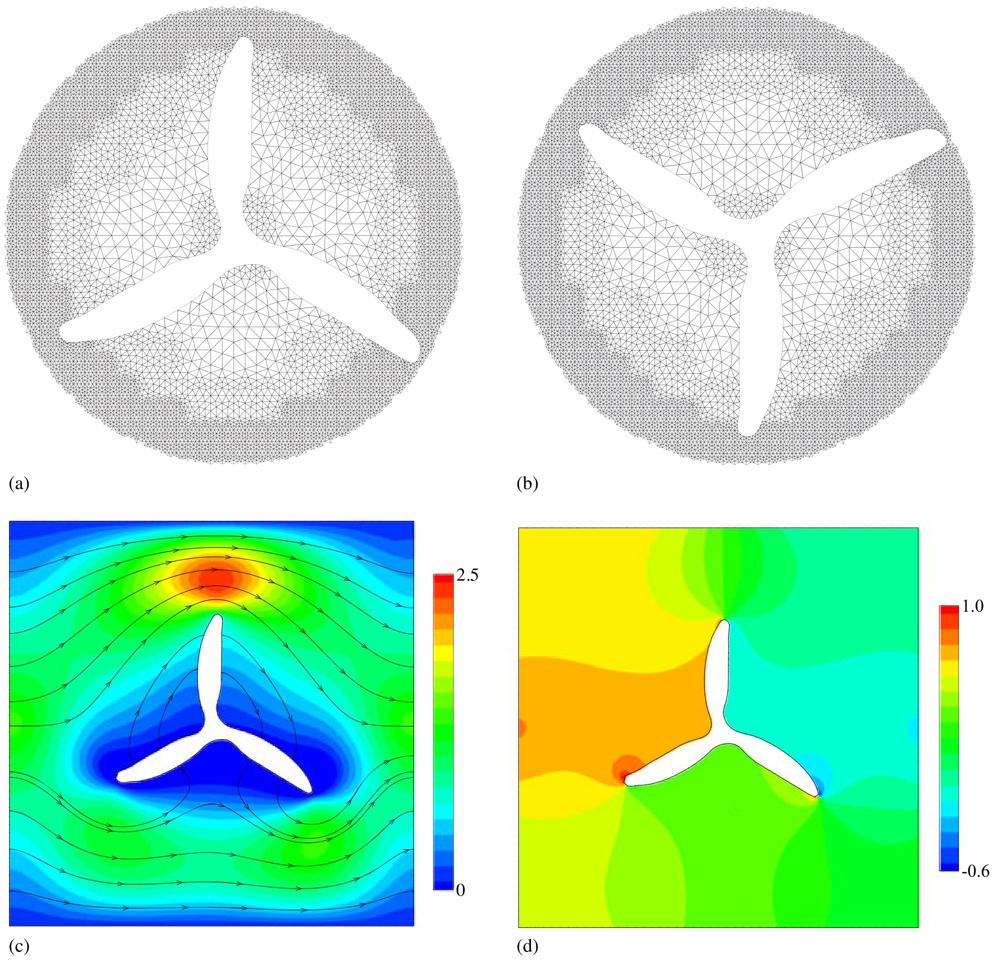


Figure 14. Simulating the problem of a Stokes flow in a channel with a propeller. At each instant, the same background mesh is used to determine a curvilinear mesh that conforms exactly to the fluid domain. Figures (a) and (b) show elements close to the propeller in the resulting mesh at two distinct times. Streamlines of the flow and contours of the velocity magnitude corresponding to the propeller orientation in (a) and angular velocity $\omega = -2$ are shown in (c). Contours of pressure at the same instant are shown in (d).

while no-slip along the boundary of the propeller implies

$$u_t = r\omega e_\theta \quad \text{on } \partial P_t, \tag{54}$$

where P_t is the domain of the propeller at time t and ω is its prescribed constant angular velocity. Since (52), (53), and (54) determine the pressure only up to a constant, p_t is assigned to be zero at one point in the flow domain, i.e., we set $p_t = 0$ at $x_0 \in B \setminus P_t$.

A triangulation of B, similar to the one shown in Figure 12, serves as the universal mesh for the fluid domain $B \setminus P_t$. The mesh is refined further near the tips of the blades to resolve features of the flow there. We adopt (curved) Taylor–Hood elements for the finite element solution of (52), i.e., the element (K_c, N^2, P^2) for the velocity u_t and (K_c, N^1, P^1) for the pressure p_t. See [52] for a discussion on the Taylor–Hood element and for the weak form of this problem, which we have omitted here. Dirichlet boundary conditions are imposed by interpolating (53) and (54) at the nodes lying on the boundary. Figures 14a and 14b show the curvilinear mesh conforming to the fluid domain at two different time instants. Since the mesh is quite refined, only the elements near the propeller are shown. Figures 14c and 14d show contours of the velocity magnitude, streamlines for the flow and the pressure computed with $\mu = 0.01$ and $\omega = -2$ at the propeller orientation shown in Figure 14a.

6.3. Application: flow interaction with a rigid disc

In the second example, we consider the interaction between a fluid and a rigid solid. The problem is to determine the trajectory of a rigid disc D of radius R and mass m immersed in an incompressible, viscous fluid flowing through a square-shaped channel B of side L. The disc is attached to the origin O located at the mid-point of the left end of the channel by a linear spring with spring constant k and equilibrium length ℓ_0. The problem setup is shown in Figure 15.

We assume that the kinematics of the fluid is governed by the equations for Stokes flow given in (52), retaining the notation introduced there. In a Cartesian coordinate system centered at O, inflow and outflow boundary conditions are prescribed at the two ends of the flow channel as

$$u_t = (L - 2|y|)e_x \quad \text{if } x = 0, L. \tag{55}$$

Denote the position of the center of the disc at time t by $c(t)$ and the disc centered at $c(t)$ by D_t. No-slip boundary conditions along the horizontal walls of the channel and along the boundary of the disc imply

$$u_t = \begin{cases} 0 & \text{if } |y| = L/2, \\ \dot{c}(t) & \text{on } \partial D_t. \end{cases} \tag{56}$$

Force balance for the disc is given by

$$m\,\ddot{c} = k\left(1 - \frac{\ell_0}{|c|}\right)c + \int_{\partial D_t} \boldsymbol{\sigma}_f \cdot n_t \, ds, \tag{57}$$

where n_t is the unit outward normal to ∂D_t and the stress $\boldsymbol{\sigma}_f$ in the fluid is computed as

$$\boldsymbol{\sigma}_f = -p_t\,\mathbb{I} + \mu\left(\nabla u_t + \nabla u_t^T\right).$$

Balance equations (52), (57), boundary conditions (55), (56), initial conditions $c(0) = c_0, \dot{c}(0) = 0$, and $p_t(x_0 \in B \setminus D_t) = 0$ together constitute a coupled system of equations for the unknowns (u_t, p_t) and $c(t)$. We use (curved) Taylor–Hood elements for the flow solution as before and adopt a staggered time integration scheme. At each time instant t_n, given the center of the disc c_n and its velocity $\dot{c}_n$, we define curvilinear elements for the flow variables over $B \setminus D_t$. We then compute the flow solution (u_n^h, p_n^h) at this time. The net force on the disc is evaluated using (57). Using central differences, we update the position and velocity of the disc to the next instant t_{n+1} and repeat the process.

The background mesh shown in Figure 15a serves as a universal mesh for the flow domain $B \setminus D_t$. Figure 15b shows the trajectory determined for the disc and its final configuration computed with parameters $L = 1$ for the container, $\mu = 0.01$ for the fluid, $R = L/10, m = 1$ for the disc,

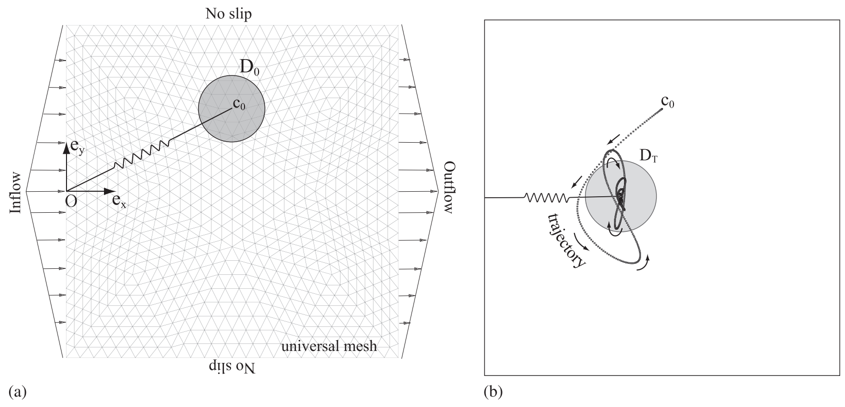


Figure 15. Initial setup for the problem of a rigid disc D interacting with an incompressible fluid is shown on the left. The disc is attached to the origin by a linear spring. Inflow and outflow boundary conditions for the flow through the channel B are indicated. A simple, unstructured mesh over B serves as the universal mesh for the fluid for the entire duration of the simulation. Figure (b) shows the trajectory computed for the disc as it moves from its initial position to an (approximate) equilibrium position.

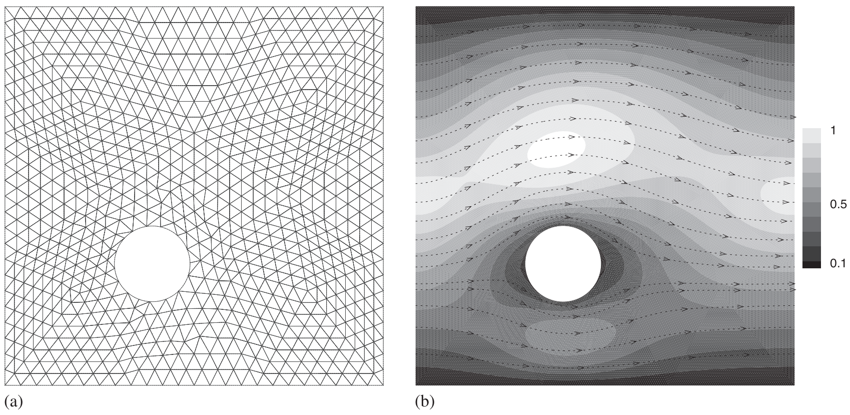


Figure 16. Conforming mesh for the fluid domain at an intermediate position of the disc (prior to reaching equilibrium) in the simulation discussed in Section 6.3 is shown on the left. Notice the curved edges along the boundary of the disc. Figure (b) shows the contours of the horizontal component of the velocity and the streamlines of the flow at the same instant.

$k = 1, \ell_0 = L/4$ for the spring, $c_0 = (0, L/4)$ for the initial position of the disc, and a time step $\Delta t = 0.05$. At an intermediate position of the disc, contours of the horizontal component of the flow velocity along with a few streamlines are shown in Figure 16. The trajectory of the disc plotted in Figure 15b shows that the disc eventually settles to an equilibrium position balancing the forces exerted by the fluid and the spring.

We refer the reader to [5, 21, 30, 53] for alternative methods to simulate such problems with changing flow domains.

7. CONCLUDING REMARKS

The meshing algorithm, the mappings to curvilinear triangles for constructing high-order curved finite elements, and the idea of universal meshes are useful tools for an important class of computationally challenging problems. By employing them in problems with evolving fluid domains while using a single background mesh, we demonstrated the algorithmic advantages they offer. We envision their application to a variety of problems with evolving domains, such as crack propagation, phase transformation, and complex fluid-structure interactions.

Important questions remain for these tools to also be useful in realistic engineering applications. A significant one is knowing the refinement required for the background mesh given a curved domain. Precise and computable estimates are valuable because they can help determine if and when a background(universal) mesh needs to be changed during the course of simulating an evolving domain. The analysis in [41] will be useful in determining such bounds.

A second challenge lies in relaxing the regularity required on domains, which currently stems from choosing the closest point projection to parameterize the boundary. Both the problem as well as the numerical scheme chosen to approximate its solution can result in domains with corners, cracks, and interfaces. To handle such domains, additional restrictions may have to be imposed on the background mesh. An important step in this direction is parameterizing immersed curves with end points and corners. We have shown how to do this in [39].

In Section 1, we mentioned arbitrary Lagrangian–Eulerian methods that prescribe a motion for vertices in meshes, and their difficulty with mesh distortion resulting from large motions of moving domains. Immersing moving domains in a universal mesh introduces different challenges, arising from the fact that a continuous motion of the boundary may not induce a continuous motion for its vertices. It is however conceivable that the two methods can in fact complement each other.

We think the ideas introduced here can be extended to meshing three-dimensional domains immersed in background meshes of tetrahedral meshes. An analysis will reveal the necessary requirements on the background mesh.

REFERENCES

1. Frey P, George P. *Mesh Generation*. Wiley-ISTE: New York, 2010.
2. Owen S. A survey of unstructured mesh generation technology. *7th International Meshing Roundtable*, Vol. 3, Dearborn MI, 1998; 239–267.
3. Shewchuk J. Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. *Applied Computational Geometry Towards Geometric Engineering* 1996; **1148**:203–222.
4. Shyy W, Udaykumar H, Rao M, Smith R. *Computational Fluid Dynamics with Moving Boundaries*. Dover Publications: Mineola, New York, 2007.
5. Donea J, Huerta A, Ponthot J, Rodríguez-Ferran A. Arbitrary Lagrangian–Eulerian methods. *Encyclopedia of Computational Mechanics* 2004.
6. Löhner R, Yang C. Improved ALE mesh velocities for moving bodies. *Communications in Numerical Methods in Engineering* 1998; **12**(10):599–608.
7. Stein K, Tezduyar T, Benney R. Automatic mesh update with the solid-extension mesh moving technique. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**(21):2019–2032.
8. Lesoinne M, Farhat C. Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering* 1996; **134**(1):71–90.
9. Farhat C, Geuzaine P. Design and analysis of robust ale time-integrators for the solution of unsteady flow problems on moving grids. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**(39):4073–4095.
10. Baiges J, Codina R. The fixed-mesh ALE approach applied to solid mechanics and fluid-structure interaction problems. *International Journal for Numerical Methods in Engineering* 2010; **81**(12):1529–1557.
11. Farhat C, Degand C, Koobus B, Lesoinne M. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**(1):231–245.
12. Blom F. Considerations on the spring analogy. *International Journal for Numerical Methods in Fluids* 2000; **32**(6):647–668.
13. Persson P, Peraire J. Curved mesh generation and mesh refinement using Lagrangian solid mechanics. *Proceedings of the 47th AIAA Aerospace Sciences Meeting and Exhibit*, Orlando, FL, 2009.
14. De Almeida V. Domain deformation mapping: application to variational mesh generation. *SIAM Journal on Scientific Computing* 1999; **20**(4):1252–1275.
15. Helenbrook B. Mesh deformation using the biharmonic operator. *International Journal for Numerical Methods in Engineering* 2003; **56**(7):1007–1021.
16. Budd C, Huang W, Russell R. Adaptivity with moving grids. *Acta Numerica* 2009; **18**(1):111–241.
17. Bank R, Smith R. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis* 1997; **34**(3):979–997.
18. Li R, Tang T, Zhang P. Moving mesh methods in multiple dimensions based on harmonic maps. *Journal of Computational Physics* 2001; **170**(2):562–588.
19. Thoutireddy P, Ortiz M. A variational r-adaption and shape-optimization method for finite-deformation elasticity. *International Journal for Numerical Methods in Engineering* 2004; **61**(1):1–21.
20. Zielonka M, Ortiz M, Marsden J. Variational r-adaption in elastodynamics. *International Journal for Numerical Methods in Engineering* 2008; **74**(7):1162–1197.
21. Saksono P, Dettmer W, Perić D. An adaptive remeshing strategy for flows with moving boundaries and fluid–structure interaction. *International Journal for Numerical Methods in Engineering* 2007; **71**(9):1009–1050.
22. Moumnassi M, Belouettar S, Bechet E, Bordas S, Quoirin D, Potier-Ferry M. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering* 2011; **200**(5–8):774–796.
23. Bastian P, Engwer C. An unfitted finite element method using discontinuous Galerkin. *International Journal for Numerical Methods in Engineering* 2009; **79**(12):1557–1576.
24. Angot P, Bruneau C, Fabrie P. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik* 1999; **81**(4):497–520.
25. Burman E, Hansbo P. Fictitious domain finite element methods using cut elements: I. a stabilized Lagrange multiplier method. *Computer Methods in Applied Mechanics and Engineering* 2010; **199**(41–44):2680–2686.
26. Glowinski R, Pan T, Periaux J. A fictitious domain method for Dirichlet problem and applications. *Computer Methods in Applied Mechanics and Engineering* 1994; **111**(3):283–303.
27. Codina R, Baiges J. Approximate imposition of boundary conditions in immersed boundary methods. *International Journal for Numerical Methods in Engineering* 2009; **80**(11):1379–1405.
28. Hansbo A, Hansbo P. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**(47–48):5537–5552.
29. Sukumar N, Moës N, Moran B, Belytschko T. Extended finite element method for three-dimensional crack modeling. *International Journal for Numerical Methods in Engineering* 2000; **48**(11):1549–1570.
30. Wagner G, Moës N, Liu W, Belytschko T. The extended finite element method for rigid particles in Stokes flow. *International Journal for Numerical Methods in Engineering* 2001; **51**(3):293–313.
31. Hemker P, Hoffmann W, van Raalte M. Discontinuous Galerkin discretisation with embedded boundary conditions. *Computational Methods in Applied Mathematics* 2003; **3**(1):135–158.

32. Lew AJ, Buscaglia GC. A discontinuous-Galerkin-based immersed boundary method. *International Journal for Numerical Methods in Engineering* 2008; **76**(4):427–454.
33. Cockburn B, Solano M. Solving dirichlet boundary-value problems on curved domains by extensions from subdomains. *SIAM Journal on Scientific Computing* 2012; **34**(1):497–519.
34. Boettinger W, Warren J, Beckermann C, Karma A. Phase-field simulation of solidification 1. *Annual Review of Materials Research* 2002; **32**(1):163–194.
35. Anderson D, McFadden G, Wheeler A. Diffuse-interface methods in fluid mechanics. *Annu. Rev. Fluid Mech* 1998; **30**:139–65.
36. Babuška I, Pitkäranta J. The plate paradox for hard and soft simple support. *SIAM Journal on Mathematical Analysis* 1990; **21**:551.
37. Wriggers P. *Computational Contact Mechanics*. Springer: New York, 2006.
38. Henry D. *Perturbation of the Boundary in Boundary-value Problems of Partial Differential Equations*. Cambridge University Press: Cambridge, 2005.
39. Rangarajan R, Lew A. Parameterization of planar curves immersed in triangulations with application to finite elements. *International Journal for Numerical Methods in Engineering* 2011; **88**(6):556–585.
40. Pebay P, Baker T. Analysis of triangle quality measures. *Mathematics of Computation* 2003; **72**(244):1817–1840.
41. Rangarajan R, Lew A. Analysis of a method to parameterize planar curves immersed in triangulations. *SIAM Journal on Numerical Analysis* 2013; **51**(3):1392–1420.
42. Bern M, Eppstein D, Gilbert J. Provably good mesh generation. *Journal of Computer and System Sciences* 1994; **48**(3):384–409.
43. Rangarajan R, Lew A. Optimization-based directional vertex relaxation and application to triangulation with universal meshes. unpublished.
44. Amenta N, Bern M, Eppstein D. Optimal point placement for mesh smoothing. *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, Louisiana, 1997; 528–537.
45. Hansen G, Douglass R, Zardecki A. *Mesh Enhancement: Selected Elliptic Methods, Foundations and Applications*. Imperial College Press: London, 2005.
46. Ciarlet P, Raviart P. Interpolation theory over curved elements, with applications to finite element methods. *Computer Methods in Applied Mechanics and Engineering* 1972; **1**(2):217–249.
47. Gordon W, Hall C. Transfinite element methods: blending-function interpolation over arbitrary curved element domains. *Numerische Mathematik* 1973; **21**(2):109–129.
48. Zlámal M. The finite element method in domains with curved boundaries. *International Journal for Numerical Methods in Engineering* 1973; **5**(3):367–373.
49. Mansfield L. Approximation of the boundary in the finite element solution of fourth order problems. *SIAM Journal on Numerical Analysis* 1978:568–579.
50. Ciarlet P. *The Finite Element Method for Elliptic Problems*, Vol. 4. North Holland: Amsterdam, 1978.
51. Chapelle D, Bathe K. *The Finite Element Analysis of Shells: Fundamentals*. Springer Verlag: Berlin and New York, 2003.
52. Girault V, Raviart P. *Finite Element Methods for Navier-stokes Equations: Theory and Algorithms*, Springer Series in Computational Mathematics, vol. 5. Springer-Verlag: Berlin and New York, 1986.
53. Bazilevs Y, Calo V, Hughes T, Zhang Y. Isogeometric fluid-structure interaction: theory, algorithms, and computations. *Computational Mechanics* 2008; **43**(1):3–37.
54. Balay S, Brown J, Buschelman K, Gropp W, Kaushik D, Knepley M, McInnes L, Smith B, Zhang H. PETSc Web page 2011. http://www.mcs.anl.gov/petsc.